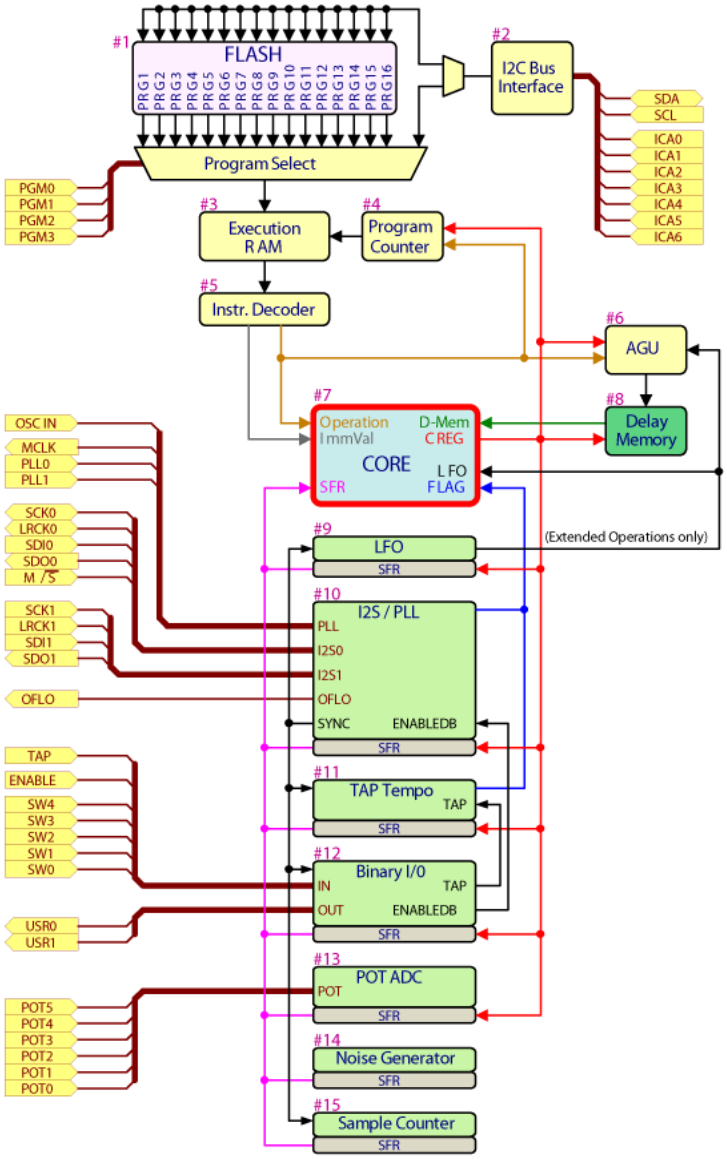
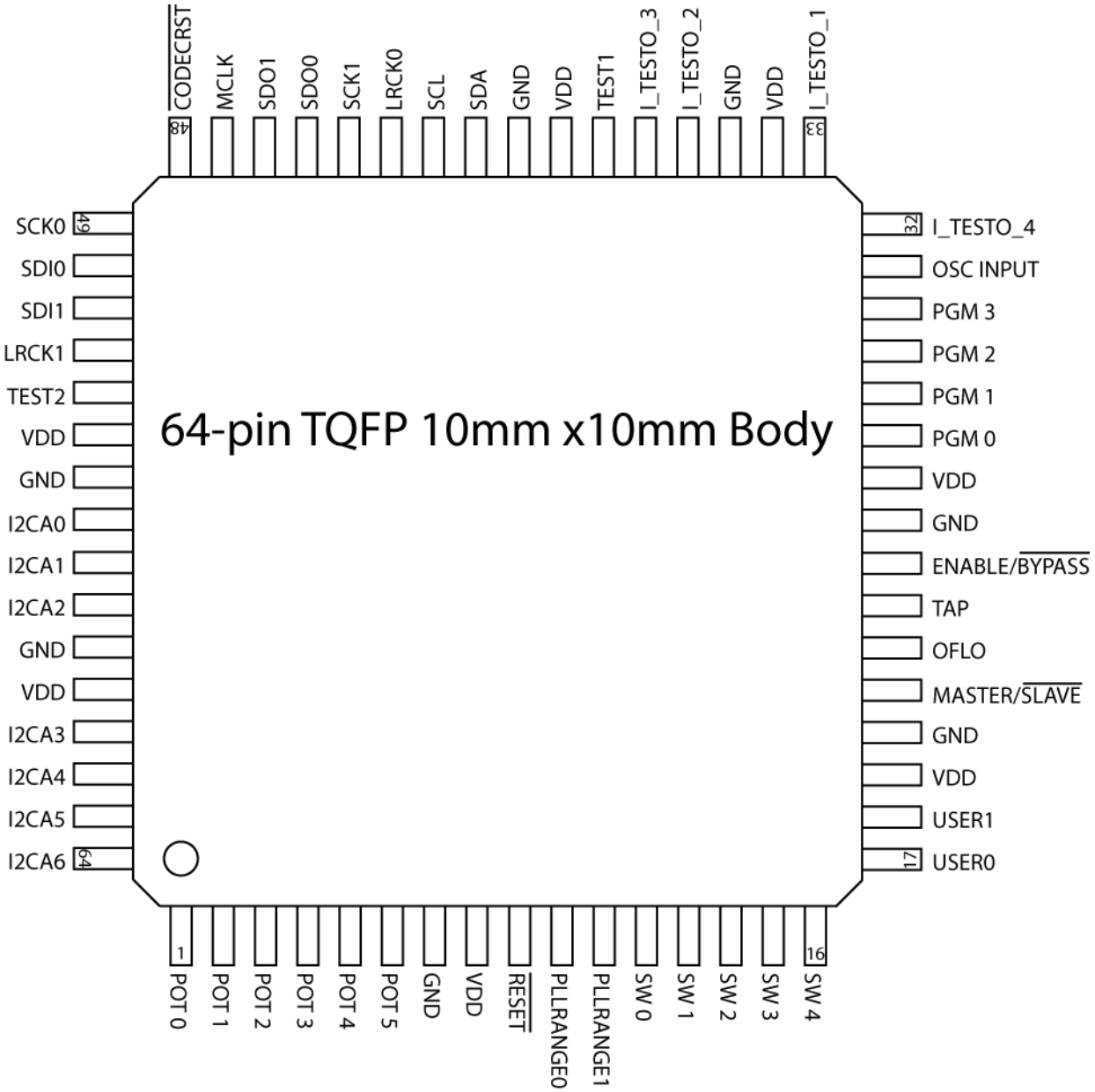
3.3V operation
32-bit data path and operations
Both saturating and non-saturating math
32-bit and 64-bit accumulators
32K x 16-bit circular delay RAM
LOG and EXP instructions
16 core registers
128 x 32-bit RAM based registers
4 digital SIN LFOs, 2 Ramp LFOs

Built in FLASH for 16 programs
I2C interface for programming FLASH
User settable I2C address via pins
2 I2S interfaces
Adjustable sample rate, master or slave selectable
6 12-bit POT inputs
5 user switch inputs, 1 tap tempo input
2 user outputs
1 enable/bypass switch input



©2020 Experimental Noize



FXCore Package

FXCore Pin List

	Function	Direction	Type	Notes
1	Pot input 0	Input	Analog	12-bit ADC potentiometer in
2	Pot input 1	Input	Analog	12-bit ADC potentiometer in
3	Pot input 2	Input	Analog	12-bit ADC potentiometer in
4	Pot input 3	Input	Analog	12-bit ADC potentiometer in
5	Pot input 4	Input	Analog	12-bit ADC potentiometer in
6	Pot input 5	Input	Analog	12-bit ADC potentiometer in
7	GND	PWR	Ground	
8	VDD	PWR	+3.3V	
9	nRESET	Input	NO PU/PD	Reset, low to reset
10	PLL Range 0	Input	Pull up	See “PLL Range Settings”
11	PLL Range 1	Input	Pull up	See “PLL Range Settings”
12	Switch 0	Input	Pull up	Switch input
13	Switch 1	Input	Pull up	Switch input
14	Switch 2	Input	Pull up	Switch input
15	Switch 3	Input	Pull up	Switch input
16	Switch 4	Input	Pull up	Switch input
17	User 0	Output	Max 10mA	User output 10mA sink/source
18	User 1	Output	Max 10mA	User output 10mA sink/source
19	VDD	PWR	+3.3V	
20	GND	PWR	Ground	
21	Master/nSlave	Input	Pull up	Master/slave select
22	Overflow LED	Output	Max 10mA	Overflow output
23	Tap Tempo input	Input	Pull up	Switch input for tap tempo
24	Enable/nBypass	Input	Pull up	Enable/bypass select
25	GND	PWR	Ground	
26	VDD	PWR	+3.3V	
27	Program 0	Input	NO PU/PD	Program select input
28	Program 1	Input	NO PU/PD	Program select input
29	Program 2	Input	NO PU/PD	Program select input
30	Program 3	Input	NO PU/PD	Program select input
31	Oscillator input	Input	NO PU/PD	High speed clock input
32	I_TESTO_4	NO CONNECT	NO CONNECT	
33	I_TESTO_1	NO CONNECT	NO CONNECT	
34	VDD	PWR	+3.3	
35	GND	PWR	Ground	
36	I_TESTO_2	NO CONNECT	NO CONNECT	
37	I_TESTO_3	NO CONNECT	NO CONNECT	

©2020 Experimental Noize

38	Test 1	NO CONNECT	NO CONNECT	
39	VDD	PWR	+3.3V	
40	GND	PWR	Ground	
41	SDA	In/Out	Open collector	I2S serial data
42	SCL	Input	Open collector	I2C serial clock
43	LRCK 0	In/Out		Output when master, input when slave
44	SCK 1	Input	NO PU/PD	Always an input, MUST tie to SCK0
45	SDO 0	Output		Data out to CODECs SDI
46	SDO 1	Output		Data out to CODECs SDI
47	MCLKOUT	In/Out	Output	Output when master, tri-state when slave
48	nCODEC Reset	Output		Output to CODEC reset
49	SCK 0	In/Out		Output when master, input when slave
50	SDI 0	Input	NO PU/PD	Data in from CODECs SDO, tie to ground if unused
51	SDI 1	Input	NO PU/PD	Data in from CODECs SDO, tie to ground if unused
52	LRCK 1	Input	NO PU/PD	Always an input, MUST tie to LRCK 0
53	Test 2	NO CONNECT	NO CONNECT	
54	VDD	PWR	+3.3V	
55	GND	PWR	Ground	
56	I2CA0	Input	Pull up	I2C address pin
57	I2CA1	Input	Pull up	I2C address pin
58	I2CA2	Input	Pull up	I2C address pin
59	GND	PWR	Ground	
60	VDD	PWR	+3.3V	
61	I2CA3	Input	Pull up	I2C address pin
62	I2CA4	Input	Pull up	I2C address pin
63	I2CA5	Input	Pull up	I2C address pin
64	I2CA6	Input	Pull up	I2C address pin

NOTE: All pins marked “NO CONNECT” MUST be left unconnected on the PCB.

©2020 Experimental Noize

Electrical Characteristics

Parameter	Min	Typ	Max	Unit
DC operating voltage (V_{DD})	3.0	3.3	3.6	V_{DC}
DC operating current[1]	-	110	-	mA
Max current sink/source through USER0, USER1 or OFLO (I_{IO})	-	-	10	mA
Operating temperature range	-30 (-22)	-	+80 (+176)	°C (°F)
Storage temperature range	-50 (-58)	-	+140 (+284)	°C (°F)
Clock range in to pin 31[2]	10	12.288	12.3	MHz
Sample rate range as master[2]	9.766	48	48.046	KHz
Sample rate range as slave[3]	0	-	96	KHz
POT0 – POT5 source impedance	-	-	10	KΩ
Internal pull-up equiv resistance[4]	5	-	50	KΩ
Input low voltage to SWO – SW4, ENABLE and TAP	GND	-	0.2* V_{DD}	V_{DC}
Input high voltage to SWO – SW4, ENABLE and TAP	0.8* V_{DD}		V_{DD}	V_{DC}
Output low voltage to USER0, USER1 and OFLO[5]	0	-	0.4	V_{DC}
Output high voltage to USER0, USER1 and OFLO[5]	2.4	-	V_{DD}	V_{DC}
Estimated FLASH endurance (Erase/Write cycles)[6]		10,000		

NOTES:

[1] : Actual current draw is dependent on program and current draw on USER0, USER1 and OFLO

[2] : The FXCore is designed to run internally at a high speed while dividing down the clock for the desired sample rate according to the PLL pin settings. If the desired sample rate cannot be accomplished by using the PLL divisor settings and keeping the source clock in the required range then the FXCore should be run in slave mode and the desired sample rate applied to it. In slave mode the PLL and sample rate clock are “decoupled”.

[3] : In slave mode the desired sample rate clock must be applied to LRCK0 and LRCK1 and the properly phase aligned bit clock to SCK0 and SCK1. The clock to SCK0/1 is 64x sample clock and must have falling edge aligned with changing LRCK0/1 clock edge.

[4] : This value is not tested in production

[5] : These values were characterized at I_{IO} = 10mA, V_{DD} = 3.3V

[6] : This is an estimate value based on process information, for development users should use FXCore Assembler and dev board which saves the flash by directly loading and running from RAM.

©2020 Experimental Noize

General Description

FXCore from Experimental Noize is a DSP designed for creating audio effects in pro-audio devices. FXCore can operate as a standalone device when paired with appropriate CODECs where it generates all necessary clocks or as part of a more complex system where it will lock to the word clock and bit clock provided by another part of the system.

The internal execution speed of the FXCore is based solely on the high speed clock applied to pin 31 OSC INPUT. When in master mode the word clock LRCK0/1 and bit clock SCK0/1 are derived from this high speed clock according to the settings of pins 10 “PLL RANGE 0” and 11 “PLL RANGE 1” as detailed later in this document. When in slave mode the word clock and bit clock pins are inputs and execution of the users program will begin on the falling edge of the word clock.

FXCore can store up to 16 programs and their associated register preset values in its internal FLASH memory. FXCore instructions are not all single clock instructions, as a result it is possible to write a program longer than there is time to execute. The FXCore assembler estimates the total core usage of your program when it is assembled and prints this information under the code in the listing file.

The FXCore was developed around a block philosophy, each block executes a specific function allowing the programmer to focus on development of their application rather than being concerned about peripheral issues such as getting digital audio data in to and out of the chip.

These blocks contain Special Function Registers (SFRs) that may be read and/or written to under program control, at program load or at boot of the FXCore depending upon the use of the SFR. Some of these SFRs set or control features of a block while other provide information about the status of blocks. For example the frequency control SFR for an LFO can be both read and written under program control as well as preset in the program header but the register containing status flags (i.e. clipping, etc.) may only be read under program control while the register for setting the debounce time of a switch may only be preset in the program header and may not be read or written by a user program.

Additionally, these blocks may contain sub-blocks that add additional functionality or resources to the block. The Core block contains a sub-block called Core Registers (CREGs) that are a set of tightly coupled register that the Arithmetic and Logic Unit (ALU) may use as source and destination registers depending on the instruction. There is also a sub-block called Memory Registers (MREGs) that are not tightly coupled to the ALU but are available to the core as additional 32-bit storage.

Block Descriptions

Block 1: FLASH

The FLASH memory block can store up to 16 programs and their associated register preset values. During normal RUN operation the programs to run is selected by the PGM03 pins and is then loaded into the Execution RAM (block 3).

Program Number - PGMX relationships

Program#	PGM3	PGM2	PGM1	PGM0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

The FLASH is arranged as 16 blocks, one for each program. Each block contains up to 1K instructions and the preset values for CREGs, MREGs and SFRs for that program. By including values for CREG, MREG and SFR registers it allows each program to preset these values without wasting instructions.

Note that the FXCore may not be able to execute all 1K instructions in a sample period, each instruction in the FXCore takes 1 or more INSCLKs (instruction clocks) to execute therefor the total number of INSCLKs a program takes to execute is what will define the maximum number of instructions, up to 1024, that may be executed in a sample period. There are approximately 3500 INSCLKs available per sample period at 48KHz, note that at lower sample rates more INSCLKS are available.

Programs must not exceed the INSCLKs available, doing so may not allow internal registers to properly transfer audio to/from the I2S lines or memory areas. The FXCore assembler will report after compilation the expected number of INSCLKs a program uses and will error out if a program attempts to exceed this number. A program which cannot complete at 48KHz sample rate may execute fine at 32KHz sample rate.

On power up, exit of programming mode or a state change on a PGMX pin in normal mode the selected program is loaded from the internal FLASH to the Execution RAM (block 3). During this loading operation the Delay Memory (block 8) is cleared, the register preset values are loaded into the target registers and the output is muted. If a user selects a program location in the FLASH that has not actually been loaded with a program or an error occurs with reading from the FLASH the FXCore will leave the outputs muted and enter an error state that can be cleared by selecting a different program. This allows an end user to recover to an operational state by simply changing the selected program.

©2020 Experimental Noize

The program select pin debounce time is set by the PRGDBRLD SFR, this debounce time is set as a number of samples. This SFR can only be set by a .SREG directive.

A list of the default programs in the FLASH is included at the end of this datasheet.

Block 2: I2C Bus Interface

Programming of the FLASH memory block is accomplished through the I2C bus. The FXCore must first be put into PROG mode and a specific set of steps must be followed to write a user program to the FLASH memory. See the document "**Communicating with the FXCore**" for details on these steps.

The ICA0 – ICA6 pins set the I2C address of the FXCore. Care should be used setting this address so as to not conflict with other devices on the bus or that may be plugged into the bus (i.e. an in circuit programmer). Do not set the address to the ranges of 0x00-0x07 or 0x78-0x7F, these are reserved addresses and many I2C devices will not respond to them.

The SDA and SCL pins are the data and clock pins for I2C communication. It is recommended that they each be pulled high with a 1K5 ohm resistor to keep them at a high level for the I2C interface and to allow easy connection of an in circuit programmer should it be required later.

Block 3: Execution RAM

While programs are stored in the FLASH memory (block 1) they must be copied to the Execution RAM as FLASH is too slow to execute from. Additionally the Execution RAM is used as a buffer for programming the FLASH memory, as a result in PROG mode programs are actually loaded into Execution RAM and may be executed rather than being written to FLASH. This allow the developer to make changes to their code without over stressing the FLASH memory. For details on this please see the document "**Communicating with the FXCore**"

Block 4: Program Counter

At the start of each sample period the program counter is rest to 0 and increments as each instruction is executed. The value can be changed based on the instruction executed, any of the jump instructions can add an offset to the counter to point to a location further down in the code.

Block 5: Instruction Decoder

The instruction decoder takes each instruction, separates it into its parts and can affect the operations in the CORE (block 7), Address Generation Unit (AGU) (block 6) and the Program Counter (block 4).

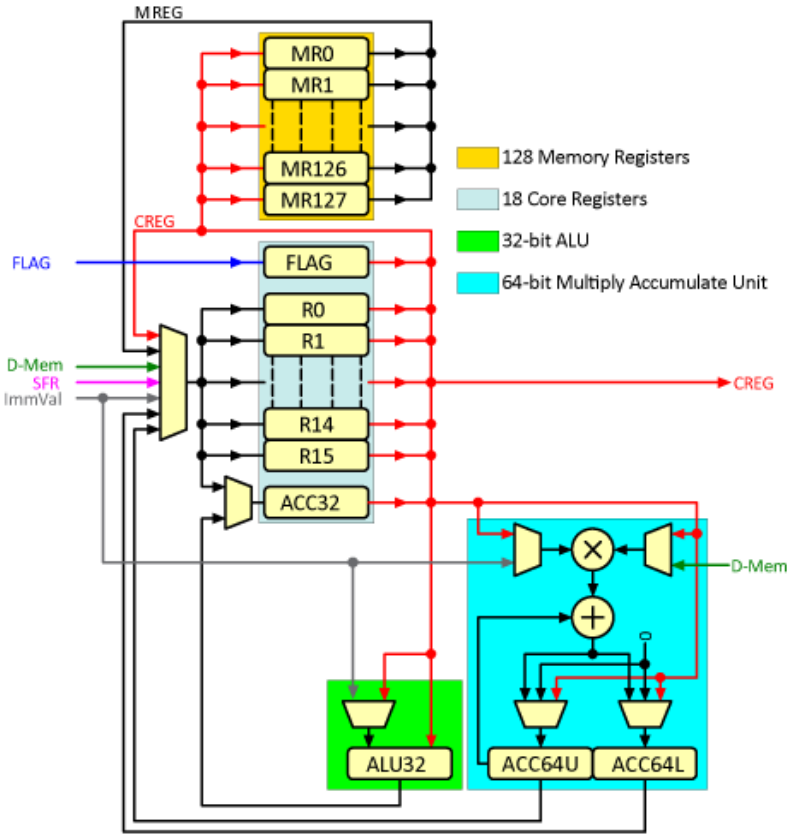
Block 6: Address Generation Unit (AGU)

The AGU generates the address for reading and writing to the Delay RAM (block 8). Typically the address

in the instruction is added to a counter in the AGU that is decremented at the start of each sample period. Some instructions such as RDDELX do not have the address in them and use a core register as a pointer into memory. Additionally some instruction such as RDDIRX both lack an explicit address and disable the addition of the counter to the pointer.

Additionally the LFOs can be added in the AGU to allow for modulating pointers to the Delay RAM as well as for pitch shifting operations.

Block 7: CORE



The CORE Block contains the Arithmetic and Logic Unit (ALU), CORE registers (CREGs), Memory registers (MREGs) and the FLAGS register.

The ALU consists of the 32-bit multiplier, adder, saturation units, logic block and the 64-bit MAC. These functions are controlled as a result of the Instruction Decoder (block 5).

The CORE is register based, in general data is moved into registers, the data is then operated on and the result placed in a register. Data in a register can be written to Delay Memory (block 8), MREGs or to another location such as an output register or other SFR.

©2020 Experimental Noize

Most instructions use two core registers as operands and place the result in ACC32. While a user is generally able to use any CREG there are a few exceptions with some of the extended operations that require data be placed in CREG R15, these instructions require more information than can be encoded in the instruction. In these cases R15 may be called PARAM0.

There are two multipliers in the CORE, the first is a 32x32 bit multiplier that takes the top 32-bits as the result. Any 2 CREGs can be used as the sources for the multiply and the result of a multiply are placed in ACC32. The result can then be moved out of ACC32 to another CREG or MREG or the top 16-bits can be written to Delay Memory (block 8). Some operations allow for use of an immediate value in the instruction.

The second multiplier is a 32x32 bit multiplier to a 64-bit accumulator for high precision MAC operations. The MAC operations can operate in two different manners, the first is treating the multiplier and multiplicand as S.31 format numbers resulting in a S.63 format product and the accumulator is treated as a S.63 format adder. This is useful for operations that you know will never exceed +/-1.0 or where clipping to such values is acceptable. In its second mode the multiplicand is shifted 3 bits right to create 3 bits of head room and the accumulator format is S3.60. This allows for some intermediate values to increase to +/-8.0 before any clipping will occur.

The 64-bit accumulator is addressable as two 32-bit registers to allow programs to move the values into 32-bit registers. This allows users to do things like save the result of a high precision filter to two MREGs between iterations of the program.

While the 32-bit ALU and 64-bit MAC are in parallel they may not operate concurrently as only one can select what registers to operate on at a time. Core (CREG) registers:

Name	Encoding	Type	Remark
R0	00000	R/W	General use
R1	00001	R/W	General use
R2	00010	R/W	General use
R3	00011	R/W	General use
R4	00100	R/W	General use
R5	00101	R/W	General use
R6	00110	R/W	General use
R7	00111	R/W	General use
R8	01000	R/W	General use
R9	01001	R/W	General use
R10	01010	R/W	General use
R11	01011	R/W	General use
R12	01100	R/W	General use
R13	01101	R/W	General use
R14	01110	R/W	General use
R15	01111	R/W	General use/PARAM0
ACC32	10000	R/W	General use/result
FLAGS	10001	R	Read only

©2020 Experimental Noize

The CORE contains the bank of 128 32-bit MREGs. While the data in the MREGs must be moved to CREGs to be operated on they are faster to access than the delay memory. They may also be used as a lookup table by an indirect access method. See the FXCore instruction set for further information.

The CORE also contains the FLAGS register which provides status information for the program to act on:

Bit	Name	Meaning
15	OUT3OFLO	Output 3 overflow
14	OUT2OFLO	Output 2 overflow
13	OUT1OFLO	Output 1 overflow
12	OUT0OFLO	Output 0 overflow
11	IN3OFLO	Input 3 clip
10	IN2OFLO	Input 2 clip
9	IN1OFLO	Input 1 clip
8	IN0OFLO	Input 0 clip
7	XXX	RESERVED
6	XXX	RESERVED
5	TB2nTB1	0: Tap button “1” event 1: Tap button “2” event
4	TAPSTKY	TAP sticky event, user has pressed the tap button for longer than TAPSTKRLD
3	NEWTT	New tap tempo value in TAPTEMPO
2	TAPRE	Tap button release event, user stopped pressing the tap button
1	TAPPE	Tap button push event, user started pressing the tap button
0	TAPDB	Debounced tap button level, 0 if pressed and 1 if not pressed

©2020 Experimental Noize

The CORE also contains a read only SFR called BOOTSTAT which contains information about the status of the FXCore. These pins and values are only read and recorded at boot.

Bit	Name	Comment
0	PLLRANGE0	The value of the PLLRANGE0 pin
1	PLLRANGE1	The value of the PLLRANGE1 pin
2	MNS	1 if in master mode, 0 if in slave mode
3	I2CA0	The value of the I2CA0 pin
4	I2CA1	The value of the I2CA1 pin
5	I2CA2	The value of the I2CA2 pin
6	I2CA3	The value of the I2CA3 pin
7	I2CA4	The value of the I2CA4 pin
8	I2CA5	The value of the I2CA5 pin
9	I2CA6	The value of the I2CA6 pin
10	-	0
11	-	0
12	-	0
13	-	0
14	-	0
15	-	0
16	PR0	1: program 0 slot contains a program, 0: program slot is empty
17	PR1	1: program 1 slot contains a program, 0: program slot is empty
18	PR2	1: program 2 slot contains a program, 0: program slot is empty
19	PR3	1: program 3 slot contains a program, 0: program slot is empty
20	PR4	1: program 4 slot contains a program, 0: program slot is empty
21	PR5	1: program 5 slot contains a program, 0: program slot is empty
22	PR6	1: program 6 slot contains a program, 0: program slot is empty
23	PR7	1: program 7 slot contains a program, 0: program slot is empty
24	PR8	1: program 8 slot contains a program, 0: program slot is empty
25	PR9	1: program 9 slot contains a program, 0: program slot is empty
26	PR10	1: program 10 slot contains a program, 0: program slot is empty
27	PR11	1: program 11 slot contains a program, 0: program slot is empty
28	PR12	1: program 12 slot contains a program, 0: program slot is empty
29	PR13	1: program 13 slot contains a program, 0: program slot is empty
30	PR14	1: program 14 slot contains a program, 0: program slot is empty
31	PR15	1: program 15 slot contains a program, 0: program slot is empty

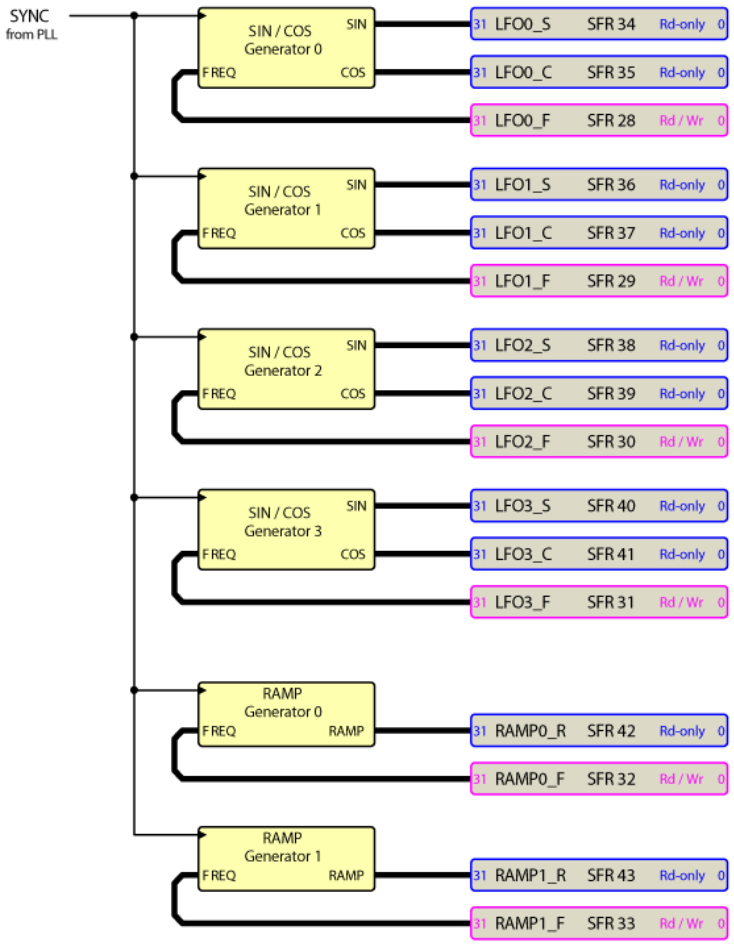
NOTE: While BOOTSTAT is a 32-bit word the mask values are 16-bit in the FXCore Assembler so the masks for things like PR0 – PR15 must be loaded as a 16-bit value and shifted up to the proper position.

Block 8: Delay Memory

The Delay Memory (block 8) consists of a 32Kx16 bank of SRAM that allows for 1 second of delay at 32KHz sample rate. This block is typically used as a rotating buffer for effects like reverb, delay, etc. but may be addressed in a direct manner for other effects.

©2020 Experimental Noize

Block 9: Low Frequency Oscillators (LFOs)



The LFO block generates low frequency SIN and COS waves for use as modulation sources for effects like chorus, phasing, etc. The block contains 4 SIN/COS LFOs and 2 RAMP generators for pitch shifting.

Each LFO has an associated special function register (SFR) to control its frequency, these SFRs are LFO0_F, LFO1_F, LFO2_F and LFO3_F. To calculate the value to write to an LFOX_F SFR to cause the associated LFO to generate the SIN and COS waves at a specific frequency use:

$C = (2^{31} - 1) * (2*pi*F)/Fs$
F = Target frequency in hertz
Fs = Sample rate
pi = 3.14159…

Each LFO produces 2 outputs: SIN and COS. These values may be read from the LFOX_S and LFOX_C (X is 0 – 3) SFRs to read the SIN and COS values respectively. These are 32-bit values and range from -1.0 to +1.0.

©2020 Experimental Noize

These LFO generators are designed to create SIN and COS waveforms in the 0 to 20Hz range. Using them to generate higher frequencies will introduce distortion into the resulting waveform.

The LFOs are used in instructions like CHR to create a chorus effect on a delay line in the Delay Memory (block 8). When used in the CHR instruction the LFO is automatically scaled to be between 0 and 1.0 so it can be multiplied by the length of the delay line then added to the base address to generate the address to read from. This has a specific advantage over older methods which used a +/- LFO value and used the center of the delay line as the base address to oscillate about. In the older method if the LFO was scaled down within the delay line the LFO still oscillated about the center of the delay, this introduced a delay between the head of the delay and the actual portion of the delay being chorused. In this newer implementation the LFO always choruses from the head of the delay so as it is scaled down no delay is introduced.

Each RAMP generator has an associated SFR to set the frequency of the RAMP. These SFRs are RAMP0_F and RAMP1_F. Calculating the coefficient for the RAMP generators is a little more complex than for the LFOs as the RAMP generators are used to shift the pitch of a signal and this depends on both the length of the delay line and if the signal is being pitched up or down. To simplify this we base the equations on a selection of four delay lengths of 512, 1024, 2048 or 4096:

To pitch up we set the coefficient to the ramp as:
$C = -2^{23} * (2^N - 1)*(512/L)$
N = Desired amount of pitch shift in octaves
L = Length of delay line used (512, 1024, 2048 or 4096)

For pitching down it is:
$C = 2^{23} * (1-(1/2^N))*(512/L)$
N = Desired amount of pitch shift in octaves
L = Length of delay line used (512, 1024, 2048 or 4096)

If the user wishes to use the ramp generator to create and use a ramp wave in their program directly the coefficient for a specific frequency of ramp can be calculated by:

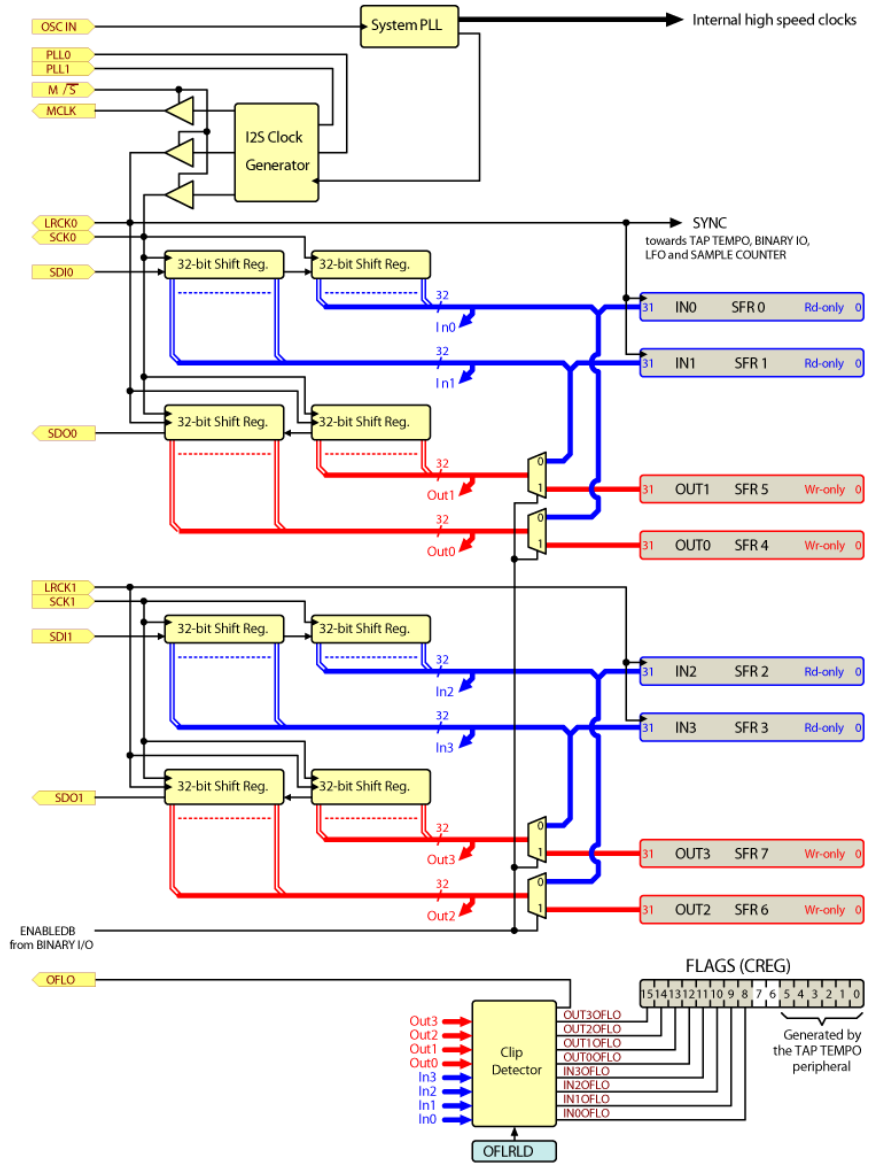
$C = (f/Fs) * 2^{32}$
f = Desired ramp frequency
Fs = Sample rate

The RAMP values can be read by reading the RAMP0_R and RAMP1_R SRFs.

All LFO and RAMP SFRs (LFOX_S, LFOX_C, RAMPX_R) are updated each sample period and as a result maintain the same value for the duration of a user's program.

©2020 Experimental Noize

Block 10: I2S and PLL



PLL Range Settings

When operating standalone or as the master clock device the FXCore can operate at different sample rates so you can select the one that is best for your project. “Clock” refers to the clock applied to pin 31 “OSC input”, setting pins 10 “PLL RANGE 0” and 11 “PLL RANGE 1” as specified in the below table will select the associated sample rate LRCK when in master mode where pin 21 “Master/nSlave” is set high. When pin 21 is set low the FXCore is locking to an external master word clock and pins 10 and 11 have no effect on sample rate.

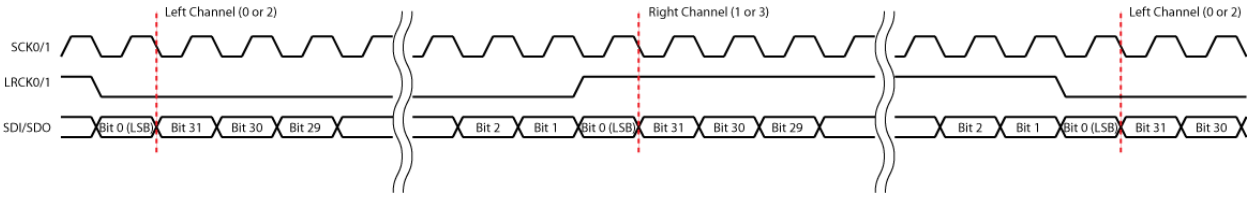
©2020 Experimental Noize

PLL Range 1 Pin 11	PLL Range 0 Pin 10	Pin 31 Clock = 12.288MHz	Pin 31 Clock = 11.2896MHz
0	0	MCLK = 3.072MHz SCK = 768KHz LRCK = 12KHz	MCLK = 2.8224MHz SCK = 705.6KHz LRCK = 11.025KHz
0	1	MCLK = 6.144MHz SCK = 1.536MHz LRCK = 24KHz	MCLK = 5.6448MHz SCK = 1.4112MHz LRCK = 22.05KHz
1	0	MCLK = 8.192MHz SCK = 2.048MHz LRCK = 32KHz	MCLK = 7.5264MHz SCK = 1.8816MHz LRCK = 29.4KHz
1	1	MCLK = 12.288MHz SCK = 3.072MHz LRCK = 48KHz	MCLK = 11.2896MHz SCK = 2.8224MHz LRCK = 44.1KHz

Generated MCLK, SCK and LRCK when FXCore is in master mode and the specified clock is applied to pin 31.

I2S Digital Audio Interfaces

The FXCore provides two standard audio interfaces allowing for up to four input and 4 output channels. Interface 0 for channels 0 and 1 will run as either a master or slave depending on the state of pin 21, interface 1 for channels 2 and 3 is always a slave and its word clock (LRCK1) and bit clock (SCK1) lines should be connected to interface 0's word clock (LRCK0) and bit clock (SCK0) lines respectively. The SDO lines of the sources (i.e. CODECs) should be connected to the SDI lines of the FXCore and the SDI lines of the sources should be connected to the SDO lines of the FXCore. The timing relationship between LRCK, SCK and SDI/SDO is illustrated in the following diagram:



Note that in master mode the exact timing will depend on the clock being supplied to pin 31 and the PLL0/1 settings. In slave mode the timing is defined by the system master. In both master and slave mode both I2S interfaces must be phase aligned, as the second I2S is always in slave mode LRCK1 should always be connected to LRCK0 and SCK1 should always be connected to SCK0.

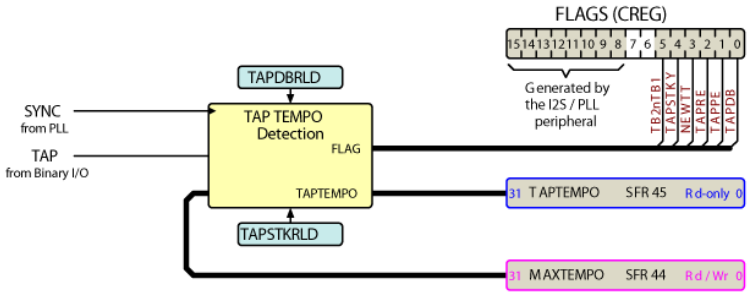
The inputs can be read in an effects program by reading the special function registers IN0 to IN3, writing to the outputs is similarly done by writing to the special function registers OUT0 to OUT3.

If any of the inputs or outputs are within 0.5db of full-scale then the corresponding bit in the CORE (block 7) FLAGS register will be set and the OFLO LED will turn on. The minimum on time for the OFLO LED is controlled by the SFR OFLORLD and is the minimum on time in samples. OFLORLD may only be set using a .SREG directive.

The FXCore uses the full 32-bits for each channel on the I2S interface.

©2020 Experimental Noize

Block 11: TAP Tempo



The TAP pin is the input for a tap tempo button. A normally open push button switch that connects to ground when pushed should be used here, the TAP pin has a weak pull-up built into the pin.

The tap tempo block inside the FXCore will handle all aspects of debounce, counting samples between presses, etc.

Typically the TAP Tempo (block 11) counts the number of samples between successive taps. If a user enters a first tap and waits longer than the time out period to enter a second tap the block will reset its self. In addition to reading the number of samples between taps the program can also determine which tap has occurred, if there is a new value and if the user is holding the tap switch closed for some period of samples known as ‘sticky”. These items are read from or controlled by a number of different SFRs:

TAPTEMPO : The number of sample periods between two successive taps.
MAXTEMPO : The maximum number of samples allowed between 2 successive taps before the block will reset its self.
TAPSTKRLD : The number of samples the user must push a tap switch for to register the switch as “sticky”. This SFR can only be set with a .SREG statement.
TAPDBRLD : The number of samples to debounce the tap input for. This SFR can only be set with a .SREG statement.

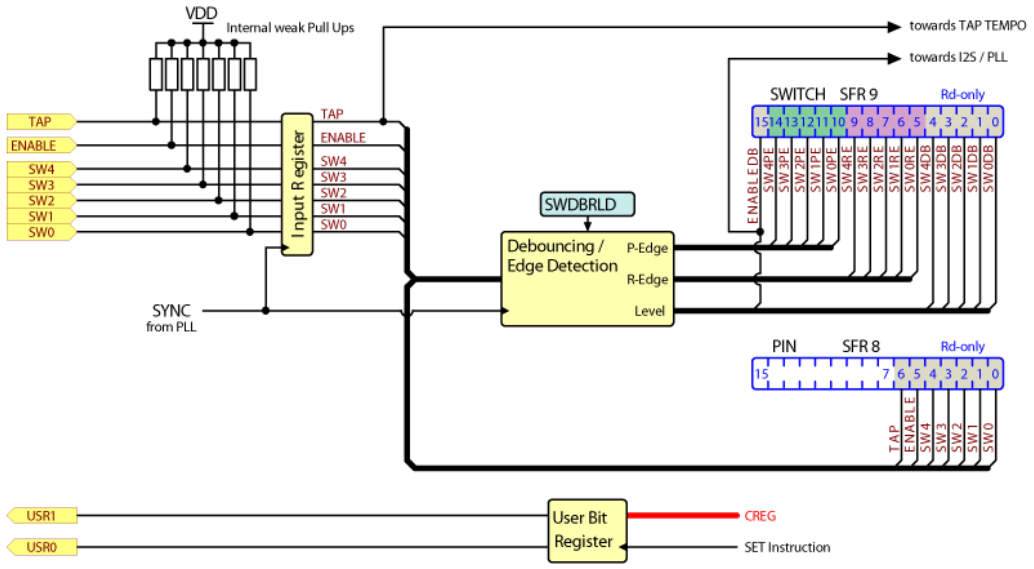
In addition to the above SFRs the tap block controls bits 0 to 5 in the FLAGS register in the CORE (block 7). Please see **Block 7: CORE** above for information on these bits.

For a more information and example code see:

“Tap Tempo Operation” section of the FXCore Instruction Set
Application note AN-4 “Using the Tap Tempo”
Default program 4 “Delay”

©2020 Experimental Noize

Block 12: Binary I/O



Pins 12 to 16, SW0 – SW4, are for designers to wire switches to that can be read during program execution. These pins have a weak internal pull-up so that an unconnected pin will be read as a “1” in the code. Pins are sampled at the sample rate and the bits in the PIN SFR are updated accordingly.

There are two ways to read the switch inputs: a raw read and a debounced read.

To read the raw pin the user reads the PIN special function register. These values are not debounced so care should be taken using them.

Debounced pin values can be read from the SWITCH special function register. The debounce time can be set in number of samples by setting the value in the SWDBRLD special function register. In addition to the debounced momentary values of SW0DB – SW4DB, SWITCH also contains two additional bit fields: SW0RE – SW4RE and SW0PE – SW4PE, with the postfixes RE and PE standing for “Release Edge” and “Push Edge”. Whenever a switch is being closed (that is the SWx input pin is being shorted to GND) the corresponding SWxPE bit within SWITCH becomes set for exactly one sample period. The SWxRE bit field indicates the opposite, SWxRE will be set upon opening the switch connected to SWx.

The “Release Edge” and “Push Edge” bit fields are useful for of detecting changes at the SWx inputs, specifically when being used in combination with momentary switches.

NOTE: SWDBRLD can only be set by a .sreg assembler directive, it cannot be set or read from the user program.

©2020 Experimental Noize

For the PIN SFR (raw level) the bits are:

Bit#	Name	Description
0	SW0	Switch 0
1	SW1	Switch 1
2	SW2	Switch 2
3	SW3	Switch 3
4	SW4	Switch 4
5	ENABLE	State of the ENABLE pin
6	TAP	State of the TAP input
7	-	Always 0
8	-	Always 0
9	-	Always 0
10	-	Always 0
11	-	Always 0
12	-	Always 0
13	-	Always 0
14	-	Always 0
15	-	Always 0

For the SWITCH SFR (debounced) the bits are:

Bit#	Name	Description
0	SW0DB	Switch 0 debounced level
1	SW1DB	Switch 1 debounced level
2	SW2DB	Switch 2 debounced level
3	SW3DB	Switch 3 debounced level
4	SW4DB	Switch 4 debounced level
5	SW0RE	High for 1 period when SW0 is released
6	SW1RE	High for 1 period when SW1 is released
7	SW2RE	High for 1 period when SW2 is released
8	SW3RE	High for 1 period when SW3 is released
9	SW4RE	High for 1 period when SW4 is released
10	SW0PE	High for 1 period when SW0 is pressed
11	SW1PE	High for 1 period when SW1 is pressed
12	SW2PE	High for 1 period when SW2 is pressed
13	SW3PE	High for 1 period when SW3 is pressed
14	SW4PE	High for 1 period when SW4 is pressed
15	ENABLEDB	ENABLE pin debounced level

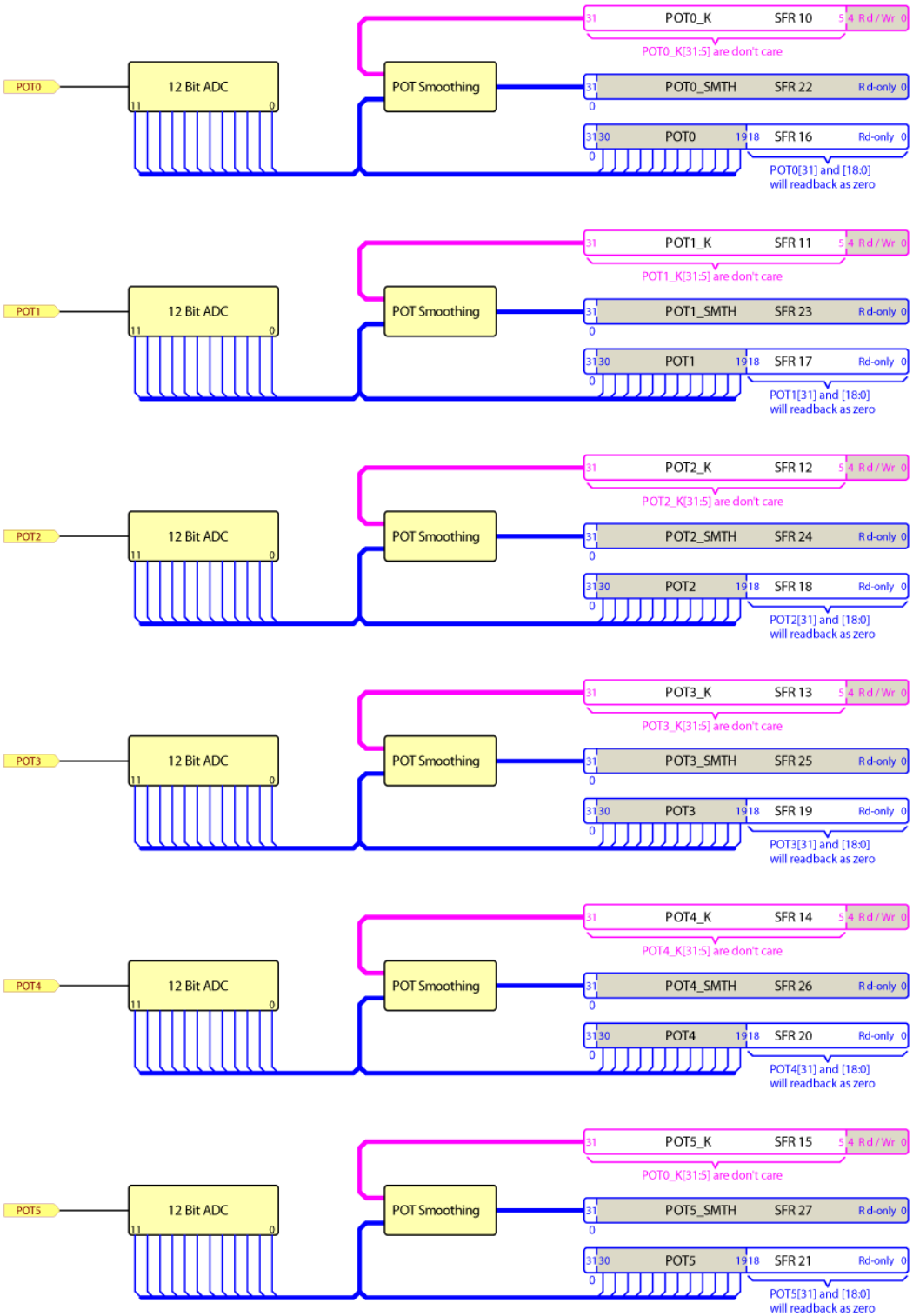
The PIN and SWITCH SFRs are updated each sample period and as a result maintain the same value for the duration of a user's program.

©2020 Experimental Noize

USER0 and USER1
The USER0 and 1 pins allow a program to give feedback on the internal status of the program such as blink at an LFO rate or control other parts of the design such as gain stages. These pins are written to by use of the SET instruction.

These pins can sink or source a maximum of 10mA.

Block 13: POT ADC



©2020 Experimental Noize

Pins 1 to 6 are 12-bit ADC inputs for real time parameter control of the effects program. Normally these pins are connected to the wiper of a potentiometer with one end of the potentiometer connected to +3.3V and the other to ground. Potentiometers should not exceed 10Kohms and a small capacitor in the 0.1uf range should be connected between the wiper and ground to aid in removing noise.

There are two ways to read each potentiometer: a raw read and a filtered read.

Reading the raw value of a potentiometer is done by reading the special function registers POT0 to POT5. This is the fastest way to read a value on the POTX pin but the user may see jitter in the value due to the lack of filtering.

Reading the filtered or “smoothed” value from the POT0_SMTH to POT5_SMTH special function registers will result in a more consistent read but may add some delay between a change in the voltage applied to the POTX pin and when it is reflected in the value read. To account for this users may change the coefficient used in the smoothing filter for each potentiometer by writing to the 5-bit special function registers POT0_K to POT5_K. The filter equation is:

POTX_SMTH = ((POTX – POTX_SMTH) >> POTX_K) + POTX_SMTH (Calculated each sample period)

This is a digital low pass filter and the 5-bit coefficient POTX_K is the number of bits the difference between the read value and the currently smoothed value is shifted. The larger the POTX_K value the longer it will take for the low pass filter to settle.

All POT SFRs and their smoothing are updated each sample period and as a result maintain the same value for the duration of a user’s program.

POT inputs are sampled at approximately 2KHz.

Block 14: Noise Generator

This block is a random number generator using the thermal noise of the FXCore to produce 32-bit values. Programs may read the current value by reading the NOISE SFR.

Block 15: Sample Counter

This block is an unsigned 32-bit counter incremented each sample period. It is reset to 0 on reset or program change. It may be read by reading from the SAMPLECNT SFR.

Default Programs

Prg#	Name	POT0	POT1	POT2	POT3	POT4	POT5	SW0	SW1	SW2	SW3	SW4	TAP
0	Small reverb	Reverb time	Diffusion	LP filter	Reverb level								
1	Medium reverb	Pre-delay	Reverb time	Damping	Reverb level								
2	Cathedral reverb	Pre-delay	Reverb time	LP filter	Reverb level			0 : Mono in use in0 (left) for both channels 1 : Stereo in					
3	Plate reverb	Reverb time	LP filter	Ringing	Reverb level								
4	Delay	delay time in pot mode	Feedback	LP Filter				Set time divide	Set time divide				Tap to set delay or hold to swotch to POT
5	Chorus	Rate	Depth	Level									
6	Flanger	Speed out	Speed back	Feedback	Level								Press for flanger to go out, release to come back
7	Through zero flanger	Speed out	Speed back	Feedback	Level	Zero point							Press for flanger to go out, release to come back
8	Phaser	Rate	Sweep range	Resonance	Depth			Set number of stages	Set number of stages				
9	Auto-wah	Sensitivity	Q										

©2020 Experimental Noize

10	Function generator	Frequency	SINE amplitude	Sawtooth amplitude	Triangle amplitude	Square wave amplitude							
11	Compander	Compression threshold	Compression ratio	Expansion threshold	Expansion ratio	Attack time	Release time	0: Compare in0 to in1 and use the higher amplitude signal in the compander on both channels 1: In0 is master and used in the compander, gain result used on both channels					
12	Rotary + reverb	Slow speed/ Reverb time	Fast speed/ Reverb diffusion	Acceleration/ LP filter	Horn-drum balance/ Reverb level	Mic distance		Select if POTs control rotary or reverb parameters	Select slow or fast rotary speed				
13	Double pitch shift	Shifter 0 range	Shifter 0 level	Shifter 1 range	Shifter 1 level	Dry level							
14	Distortion	Input gain	LP frequency	LP Q	Output level								
15	Nano looper							Playback sample forward or reverse					Press to record, release to play

©2020 Experimental Noize

Notes:

©2020 Experimental Noize

Experimental Noize Inc. reserves the right to make changes to, or to discontinue availability of, any product or service without notice.

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

Experimental Noize Inc. make no warranty, expressed or implied, of the fitness of any product or service for any particular application.

In no even shall Experimental Noize Inc. be liable for any direct, indirect, consequential, punitive, special or incidental damages including, without limitation, damages for loss and profits, business interruption, or loss of information arising out of the use or inability to use any product or document, even if Experimental Noize Inc. has been advised of the possibility of such damage.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Experimental Noize Inc. products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death (“Safety-Critical Applications”). Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Experimental Noize Inc. products are not designed nor intended for use in military or aerospace applications or environments. Experimental Noize Inc. products are not designed nor intended for use in automotive applications.

Experimental Noize Inc.

Scottsdale, AZ USA

www.xnoize.com

sales@xnoize.com