PARALLAX SEMICONDUCTOR

www.parallaxsemiconductor.com
sales@parallaxsemiconductor.com
support@parallaxsemiconductor.com
phone: 916-632-4664 • fax:916-624-8003

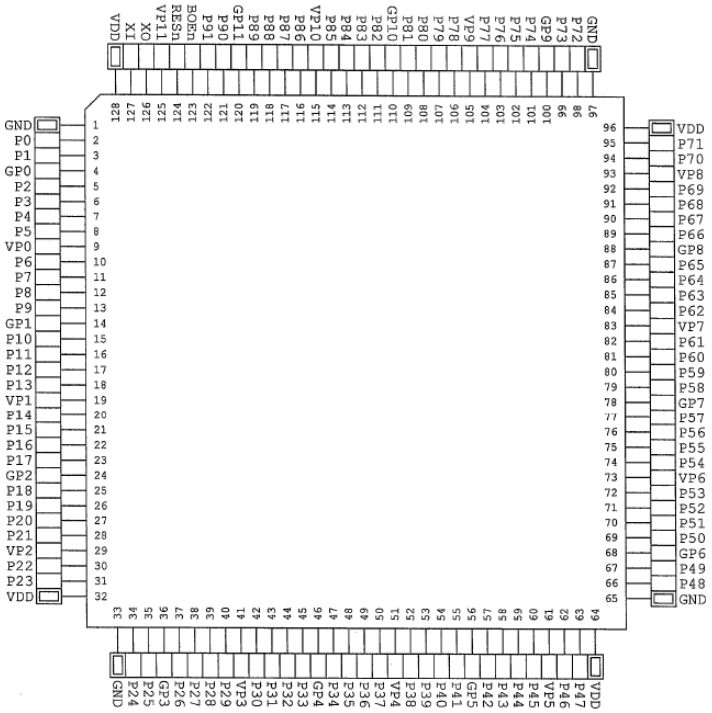
Propeller 2 Detailed Preliminary Feature List

The information in this document is preliminary and, like the Propeller 2 itself, is subject to change without notice. The information here is not guaranteed to be final, accurate, or complete.

Introduction

The Propeller 2 will be a general-purpose 32-bit microcontroller with 8 identical processors called "cogs." Each cog has 512 longs (2 KB) of memory from which it executes instructions from. All of the cogs share a 128-bit data bus to a central "hub" which will have over 128 KB of RAM+ROM (Expected = 192 KB of RAM + 2 KB of ROM). Each cog will execute one instruction per clock cycle and when accessing the hub will have to wait 0 to 7 clock cycles before being able to execute its instruction. Additionally, on system startup the chip will use a protected key that is unique to each chip to encrypt and decrypt external program storage EEPROM/FLASH. The boot loader that accesses this fuse-bit programmed key and runs its program from ROM will be in a non-user accessible part of the chip (The 2 KB of ROM will not be readable by user programs, nor will the fuse-bit key).

Figure 1: Planned 128-Pin SMT Package



- 1.8 V Core – 3.3/1.8 V I/O pins. (Each group of 8 I/O pins is powered by a VP pin and GP).
- The chip will feature a 20 KHz and 20 MHz RC oscillator.
- The chip will be able to run off an external oscillator or crystal. An internal PLL will multiply the input frequency by 1X/2X/4X/8X/16X.
- Each I/O pin is expected to be able to source or sink 40 mA.
- The chip is expected to be able to be clocked at 160 MHz in normal operation.

Hub

Each cog now features two pointer registers called PTRA and PTRB and a 16-byte/8-word/4-long read cache. The register pointers can be used for any hub memory read or write operation. They feature auto incrementing and decrementing with pre or post operation.

Table 1: Hub Memory Access Instructions

I=Immediate (0=S Mode, 1=#n/PTR Mode) – "1" in "1SUPIIIII" selects between #n/PTR
S=Select (0=PTRA, 1=PTRB), U=Update (0=No, 1=Yes), P=Pre/Post (0=Pre, 1=Post)
IIIII=Offset (-16, -15, …, 0, 1, …, 14, 15) – Shifted Left By Context (BYTE/WORD/LONG)

Machine Code	Mnemonic	Operand	Operation
000000_zc0_i_cccc_ddddddddd_1supiiiii	WRBYTE	D,S/#n/PTR	Write "bits 0-7" in register "D (0-511)" to hub memory byte address in register "S (0-511)" or address "n (0-255)" or address in PTRA/PTRB with a byte offset between (-16 to 15) or address in PTRA/PTRB with a post or pre increment byte offset of (-16 to 15). Waits between 0-7 cycles due to hub.
000000_z01_i_cccc_ddddddddd_1supiiiii	RDBYTE	D,S/#n/PTR	Read hub memory byte address in register "S (0-511)" or address "n (0-255)" or address in PTRA/PTRB with a byte offset between (-16 to 15) or address in PTRA/PTRB with a post or pre increment byte offset of (-16 to 15) to "bits 0-7" in register "D (0-511)". Waits between 0-7 cycles due to hub.
000000_z11_i_cccc_ddddddddd_1supiiiii	RDBYTEC	D,S/#n/PTR	Read hub memory byte address in register "S (0-511)" or address "n (0-255)" or address in PTRA/PTRB with a byte offset between (-16 to 15) or address in PTRA/PTRB with a post or pre increment byte offset of (-16 to 15) to "bits 0-7" in register "D (0-511)". Waits between 0-7 cycles due to hub if a cache miss occurs. Caches 16 bytes if a cache miss occurs (128-bit Bus). Waits 0 cycles on cache hit. NOTE: Cache may be outdated!!!
000001_zc0_i_cccc_ddddddddd_1supiiiii	WRWORD	D,S/#n/PTR	Write "bits 0-15" in register "D (0-511)" to hub memory word address in register "S (0-511)" or address "n (0-255)" or address in PTRA/PTRB with a word offset between (-16 to 15) or address in PTRA/PTRB with a post or pre increment word offset of (-16 to 15). Waits between 0-7 cycles due to hub.
000001_z01_i_cccc_ddddddddd_1supiiiii	RDWORD	D,S/#n/PTR	Read hub memory word address in register "S (0-511)" or address "n (0-255)" or address in PTRA/PTRB with a word offset between (-16 to 15) or address in PTRA/PTRB with a post or pre increment word offset of (-16 to 15) to "bits 0-15" in register "D (0-511)". Waits between 0-7 cycles due to hub.
000001_z11_i_cccc_ddddddddd_1supiiiii	RDWORDC	D,S/#n/PTR	Read hub memory word address in register "S (0-511)" or address "n (0-255)" or address in PTRA/PTRB with a word offset between (-16 to 15) or address in PTRA/PTRB with a post or pre increment word offset of (-16 to 15) to "bits 0-15" in register "D (0-511)". Waits between 0-7 cycles due to hub if a cache miss occurs. Caches 8 words if a cache miss occurs (128-bit Bus). Waits 0 cycles on cache hit. NOTE: Cache may be outdated!!!
000010_zc0_i_cccc_ddddddddd_1supiiiii	WRLONG	D,S/#n/PTR	Write "bits 0-31" in register "D (0-511)" to hub memory long address in register "S (0-511)" or address "n (0-255)" or address in PTRA/PTRB with a long offset between (-16 to 15) or address in PTRA/PTRB with a post or pre increment long offset of (-16 to 15). Waits between 0-7 cycles due to hub.

Machine Code	Mnemonic	Operand	Operation
000010_z01_i_cccc_ddddddddd_1supiiiii	RDLONG	D,S/#n/PTR	Read hub memory long address in register "S (0-511)" or address "n (0-255)" or address in PTRA/PTRB with a long offset between (-16 to 15) or address in PTRA/PTRB with a post or pre increment long offset of (-16 to 15) to "bits 0-31" in register "D (0-511)". Waits between 0-7 cycles due to hub.
000010_z11_i_cccc_ddddddddd_1supiiiii	RDLONGC	D,S/#n/PTR	Read hub memory long address in register "S (0-511)" or address "n (0-255)" or address in PTRA/PTRB with a long offset between (-16 to 15) or address in PTRA/PTRB with a post or pre increment long offset of (-16 to 15) to "bits 0-31" in register "D (0-511)". Waits between 0-7 cycles due to hub if a cache miss occurs. Caches 4 longs if a cache miss occurs (128-bit Bus). Waits 0 cycles on cache hit. NOTE: Cache may be outdated!!!
000011_zcr_1_cccc_000000000_000001100	CACHEX		The next read cache instruction misses causing the next read cache instruction to fill the cache with updated data.

Table 2: PTR Access Instructions

machine code	Mnemonic	Operand	Operation
000011_zcr_1_cccc_ddddddddd_000010010	GETPTRA	D	Stores PTRA in register "D (0-511)".
000011_zcr_1_cccc_ddddddddd_000010011	GETPTRB	D	Stores PTRB in register "D (0-511)".
000011_zcn_1_cccc_nnnnnnnnn_010110010	SETPTRA	D/#n	Set PTRA to register "D (0-511)" or "n (0-511)".
000011_zcn_1_cccc_nnnnnnnnn_010110011	SETPTRB	D/#n	Set PTRB to register "D (0-511)" or "n (0-511)".
000011_zcn_1_cccc_nnnnnnnnn_010110100	ADDPTRA	D/#n	Add to PTRA register "D (0-511)" or "n (0-511)".
000011_zcn_1_cccc_nnnnnnnnn_010110101	ADDPTRB	D/#n	Add to PTRB register "D (0-511)" or "n (0-511)".
000011_zcn_1_cccc_nnnnnnnnn_010110110	SUBPTRA	D/#n	Subtract from PTRA register "D (0-511)" or "n (0-511)".
000011_zcn_1_cccc_nnnnnnnnn_010110111	SUBPTRB	D/#n	Subtract from PTRB register "D (0-511)" or "n (0-511)".

N=Number (0=D register address mode, 1=n number argument mode) – Specifies Literal
OPCODE 000011 – Extended Instruction Set from P8X32A HUBOP instruction.
CCCC=Condition Codes – Conditional Execution Based on Z and C flags.

Table 3: Quad Access Instruction

Machine Code	Mnemonic	Operand	Operation
000011_zcr_1_cccc_ddddddddd_000010001	GETTOPS	D	Store in register "D (0-511)" the top byte of each long in the quad registers. E.g. "byte15.byte11.byte7.byte3" (31…0).

Table 4: Quad Transfer Instructions

Machine Code	Mnemonic	Operand	Operation
000011_zcn_1_cccc_1supiiiii_010110000	WRQUAD	D/#n/PTR	Write from cog registers defined for quad transfer hub memory longs starting from address in register "D (0-511)" or address "n (0-255)" or address in PTRA/PTRB with a long offset between (-16 to 15) or address in PTRA/PTRB with a post or pre increment long offset of (-16 to 15). Waits between 0-7 cycles due to hub.
000011_zcn_1_cccc_1supiiiii_010110001	RDQUAD	D/#n/PTR	Read hub memory longs starting from address in register "D (0-511)" or address "n (0-255)" or address in PTRA/PTRB with a long offset between (-16 to 15) or address in PTRA/PTRB with a post or pre increment long offset of (-16 to 15) to cog registers defined for quad transfer. Waits between 0-7 cycles due to hub.
000011_zcn_1_cccc_nnnnnnndd_011100010	SETQUAD	D/#n	Define for quad transfer cog memory locations in register "D (0-511)" or "n (0-511)" where only addresses that are multiples of four are valid. SETQUAD only defines a temporary window for WRQUAD and RDQUAD.

Table 5: Hub Resource Access Instructions

Machine Code	Mnemonic	Operand	Operation
000011_zcr_0_cccc_ddddddddd_sssssssss	COGINIT	D, S	Start cog at address in register "D (0-511)" with parameter in register "S (0-511). PTRA and PTRB are init'ed to S. Waits between 0-7 cycles for to hub.
000011_zcr_1_cccc_ddddddddd_000000000	CLKSET	D	Change the system clock mode according to register "D (0-511)". Waits between 0-7 cycles for to hub.
000011_zcr_1_cccc_ddddddddd_000000001	COGID	D	Store the cog ID of the cog executing this instruction in register "D (0-511)". Waits between 0-7 cycles for to hub.
000011_zcr_1_cccc_ddddddddd_000000010	COGINIT	D	Start cog in register "D (0-511)" according to parameter in register D. Waits between 0-7 cycles for to hub. **DO NOT USE—MAY NOT BE INCLUDED IN COMPILER.**
000011_zcr_1_cccc_ddddddddd_000000011	COGSTOP	D	Stop cog in register "D (0-511)" according to parameter in register D. Waits between 0-7 cycles for to hub.
000011_zcr_1_cccc_ddddddddd_000000100	LOCKNEW	D	Check out the lock in register "D (0-511)" from the poll of free locks to use. Waits between 0-7 cycles for to hub.
000011_zcr_1_cccc_ddddddddd_000000101	LOCKRET	D	Return the lock in register "D (0-511)" to the poll of free locks for use. Waits between 0-7 cycles for to hub.
000011_zcr_1_cccc_ddddddddd_000000110	LOCKSET	D	Set and get the previous state of the lock in register "D (0-511)" - Atomic. Waits between 0-7 cycles for to hub.
000011_zcr_1_cccc_ddddddddd_000000111	LOCKCLR	D	Clear and get the previous state of the lock in register "D (0-511)" - Atomic. Waits between 0-7 cycles for to hub.
000011_zcn_1_cccc_dddddnnnn_011100000	SETCOG	D/#n	Set the cog for COGINIT D, S according to register "D (0-511)" or "n (0-7)". Bit n[3] forces cognew op.

CLUT

Each cog now features a 128 Long Color Look Up Table (CLUT) designed for use with the video generator in each cog. While the video generator is in use each long in the CLUT holds R.G.B.A.X information for the video generator to display video with. When the video generator is not in use the CLUT may be used as a general-purpose memory scratch space, or as a 128 Long FIFO buffer, or as a call stack and evaluation stack (at the same time). The CLUT has two pointers used to index it called SPA and SPB.

Table 6: CLUT Instructions

Machine Code	Mnemonic	Operand	Operation
000011_zcr_1_cccc_ddddddddd_000011000	POPUPA	D	Store CLUT[SPA] in register "D (0-511)" and then increment SPA.
000011_zcr_1_cccc_ddddddddd_000011001	POPUPB	D	Store CLUT[SPB] in register "D (0-511)" and then increment SPB.
000011_zcr_1_cccc_ddddddddd_000011010	POPA	D	Decrement SPA and then store CLUT[SPA] in register "D (0-511)".
000011_zcr_1_cccc_ddddddddd_000011011	POPB	D	Decrement SPB and then store CLUT[SPB] in register "D (0-511)".
000011_zcr_1_cccc_000000000_000011100	RETA		Decrement SPA and then jump to instruction (CLUT[SPA] & 0x1FF). Flush pipeline before jump – results in a two-cycle loss.
000011_zcr_1_cccc_000000000_000011101	RETB		Decrement SPB and then jump to instruction (CLUT[SPB] & 0x1FF). Flush pipeline before jump – results in a two-cycle loss.
000011_zcr_1_cccc_000000000_000011110	RETAD		Decrement SPA and then jump to instruction (CLUT[SPA] & 0x1FF). Do not flush pipeline before jump – must be executed two instructions before intended jump space.
000011_zcr_1_cccc_000000000_000011111	RETBD		Decrement SPB and then jump to instruction (CLUT[SPB] & 0x1FF). Do not flush pipeline before jump – must be executed two instructions before intended jump space.
000011_zcn_1_cccc_000000000_010100010	SETSPA	D/#n	Set SPA to register "D (0-511)" or "n (0-511)".
000011_zcn_1_cccc_000000000_010100011	SETSPB	D/#n	Set SPB to register "D (0-511)" or "n (0-511)".
000011_zcn_1_cccc_000000000_010100100	ADDSPA	D/#n	Add to SPA register "D (0-511)" or "n (0-511)".
000011_zcn_1_cccc_000000000_010100101	ADDSPB	D/#n	Add to SPB register "D (0-511)" or "n (0-511)".
000011_zcn_1_cccc_000000000_010100110	SUBSPA	D/#n	Subtract from SPA register "D (0-511)" or "n (0-511)".
000011_zcn_1_cccc_000000000_010100111	SUBSPB	D/#n	Subtract from SPB register "D (0-511)" or "n (0-511)".
000011_zcn_1_cccc_000000000_010101000	PUSHDNA	D/#n	Decrement SPA and then store register "D (0 511)" in CLUT[SPA].
000011_zcn_1_cccc_000000000_010101001	PUSHDNB	D/#n	Decrement SPB and then store register "D (0-511)" in CLUT[SPB].
000011_zcn_1_cccc_000000000_010101010	PUSHA	D/#n	Store register "D (0-511)" in CLUT[SPA] and then increment SPA.
000011_zcn_1_cccc_000000000_010101011	PUSHB	D/#n	Store register "D (0-511)" in CLUT[SPB] and then increment SPB.
000011_zcn_1_cccc_000000000_010101100	CALLA	D/#n	Store the program counter (PC) in CLUT[SPA] and then increment SPA and then jump to the address in register "D (0-511)" or address "n (0-511)". Flush pipeline before jump – results in a two-cycle loss. D version doesn't flush.
000011_zcn_1_cccc_000000000_010101101	CALLB	D/#n	Store the program counter (PC) in CLUT[SPB] and then increment SPB and then jump to the address in register "D (0-511)" or address "n (0-511)". Flush pipeline before jump – results in a two-cycle loss. D version doesn't flush.
000011_zcn_1_cccc_000000000_010101110	CALLAD	D/#n	Store the program counter (PC) in CLUT[SPA] and then increment SPA and then jump to the address in register "D (0-511)" or address "n (0-511)"…
000011_zcn_1_cccc_000000000_010101111	CALLBD	D/#n	Store the program counter (PC) in CLUT[SPB] and then increment SPB and then jump to the address in register "D (0-511)" or address "n (0-511)"…
000011_zcr_1_cccc_ddddddddd_000010101	GETSPD	D	Stores ((SPA - SPB) & 0x7F) in register "D (0-511)". FOR FIFO MODE.
000011_zcr_1_cccc_ddddddddd_000010110	GETSPA	D	Stores SPA in register "D (0-511)".
000011_zcr_1_cccc_ddddddddd_000010111	GETSPB	D	Stores SPB in register "D (0-511)".

Math

Each cog now features the ability to perform 32-bit multi-cycle multiplies, 32-bit multi-cycle divides, 32-bit multi-cycle square roots, and 32-bit CORDIC transcendental operations. All of the advanced multi-cycle math operations use separate state machines that run concurrently with processor execution.

Note: The CORDIC algorithm rotates a point in the XY plane by a given angle. Look at X/Y/A results for SIN/COS/TAN/ARCSIN/ARCOS/ARCTAN values of X/Y/A.

Table 7: Math Operation Instructions

Machine Code	Mnemonic	Operand	Operation
000011_zcr_1_cccc_ddddddddd_000110000	GETMULL	D	Store the bottom 32 bits of the 32x32 bit multiply in register "D (0-511)", waits for multiply FSM if not done yet.
000011_zcr_1_cccc_ddddddddd_000110001	GETMULH	D	Store the top 32 bits of the 32x32 bit multiply in register "D (0-511)", waits for multiply FSM if not done yet.
000011_zcr_1_cccc_ddddddddd_000110010	GETDIVQ	D	Store the quotient of the divide in register "D (0-511)", waits for divide FSM if not done yet.
000011_zcr_1_cccc_ddddddddd_000110011	GETDIVR	D	Store the remainder of the divide in register "D (0-511)", waits for divide FSM if not done yet.
000011_zcr_1_cccc_ddddddddd_000110100	GETSQRT	D	Store the result of the square root in register "D (0-511)", waits for square root FSM if not done yet.
000011_zcr_1_cccc_ddddddddd_000110101	GETCORX	D	Store the result of the CORDIC X part in register "D (0-511)", waits for CORDIC FSM if not done yet.
000011_zcr_1_cccc_ddddddddd_000110110	GETCORY	D	Store the result of the CORDIC Y part in register "D (0-511)", waits for CORDIC FSM if not done yet.
000011_zcr_1_cccc_ddddddddd_000110111	GETCORZ	D	Store the result of the CORDIC A part in register "D (0-511)", waits for CORDIC FSM if not done yet.
000011_zcn_1_cccc_nnnnnnnnn_011000000	SETMULA	D/#n	Setup long A to be multiplied by long B given the value in register "D (0-511)" or number "n (0-511)". Will take 16 cycles.
000011_zcn_1_cccc_nnnnnnnnn_011000001	SETMULB	D/#n	Setup long B to be multiplied by long A given the value in register "D (0-511)" or number "n (0-511)". Starts multiply.
000011_zcn_1_cccc_nnnnnnnnn_011000010	SETDIVA	D/#n	Setup the dividend long given the value in register "D (0-511)" or number "n (0-511)". Will take 16 cycles.
000011_zcn_1_cccc_nnnnnnnnn_011000011	SETDIVB	D/#n	Setup the divisor long given the value in register "D (0-511)" or number "n (0-511)". Starts divide.
000011_zcn_1_cccc_nnnnnnnnn_011000100	SETSQRT	D/#n	Setup the operand given the value in register "D (0-511)" or number "n (0-511)". Starts square root calculation.
000011_zcn_1_cccc_nnnnnnnnn_011000101	SETCORX	D/#n	Setup the X Cartesian Coordinate given the value in register "D (0-511)" or number "n (0-511)" for CORDIC.
000011_zcn_1_cccc_nnnnnnnnn_011000110	SETCORY	D/#n	Setup the Y Cartesian Coordinate given the value in register "D (0-511)" or number "n (0-511)" for CORDIC.
000011_zcn_1_cccc_nnnnnnnnn_011000111	SETCORZ	D/#n	Setup the Angle given the value in register "D (0-511)" or number "n (0-511)" for CORDIC.
000011_zcn_1_cccc_ddddnnnnn_011001000	CORDROT	D/#n	Start the CORDIC rotation operation given the value in register "D (0-511) or "n (0-31)" iterations.
000011_zcn_1_cccc_ddddnnnnn_011001001	CORDATN	D/#n	Start the CORDIC arc tangent operation given the value in register "D (0-511) or "n (0-31)" iterations.
000011_zcn_1_cccc_ddddnnnnn_011001010	CORDEXP	D/#n	Start the CORDIC exponential operation given the value in register "D (0-511) or "n (0-31)" iterations.
000011_zcn_1_cccc_ddddnnnnn_011001011	CORDLOG	D/#n	Start the CORDIC logarithmic operation given the value in register "D (0-511) or "n (0-31)" iterations.

Miscellaneous Hardware

Each cog has a free running LFSR (Linear Feedback Shift Register) and System Counter that change every clock cycle. Accessing the LSFR taps into a sequence of 4,294,967,295 numbers that the LSFR traverses through in a pseudo random order. The System Counter counts the number of clock ticks since power up – it is a 64-bit counter, the LSFR is 32 Bits.

Table 8: System Counter Instructions

Machine Code	Mnemonic	Operand	Operation
000011_zcr_1_cccc_ddddddddd_000001111	GETCNT	D	Store the bottom 32 Bits of the System Counter (CNT) in register "D (0-511)". If executed again (no instruction in between previous execution) store the top 32 Bits of the System Counter in register "D (0-511)". If a roll over occurs between accesses TOP-1 is stored.
000011_zcr_1_cccc_ddddddddd_000010000	GETLFSR	D	Store the LSFR in register "D (0-511)".

Each cog additionally has a single cycle 16-bit hardware multiplier capable of unsigned and signed multiplications. The multiplication also adds into a 64-bit register for MAC ops.

Table 9: Multiply and Accumulate Instructions

Machine Code	Mnemonic	Operand	Operation
000100_zc0_i_cccc_ddddddddd_sssssssss	MAC	D,S	Multiply unsigned register "D (0-511)" and unsigned register "S (0-511)" or an immediate value (0-511) and add to the 64-bit accumulator.
000100_zc1_i_cccc_ddddddddd_sssssssss	MUL	D,S	Multiply unsigned register "D (0-511)" and unsigned register "S (0-511)" or an immediate value (0-511) and store in register D.
000101_zc0_i_cccc_ddddddddd_sssssssss	MACS	D,S	Multiply signed register "D (0-511)" and signed register "S (0-511)" or an immediate value (0-511) and add to the 64-bit accumulator.
000101_zc1_i_cccc_ddddddddd_sssssssss	MULS	D,S	Multiply signed register "D (0-511)" and signed register "S (0-511)" or an immediate value (0-511) and store in register D.
000011_zcr_1_cccc_000000000_000001101	CLRACC		Zero the Multiply Accumulator (ACC).
000011_zcr_1_cccc_ddddddddd_000001110	GETACC	D	Store the bottom 32 Bits of the ACC in register "D (0-511)". If executed again (no instruction in between previous execution) store the top 32 Bits of the ACC in register "D (0-511)".

Miscellaneous Instructions

Each cog additionally features a number of new instructions to make many common operations much easier to perform than before. Most of the new instructions are in the extended instruction set while a few of the new instruction are in the original set.

Table 10: Extended Miscellaneous Instructions

Machine Code	Mnemonic	Operand	Operation
000011_zcr_1_cccc_ddddddddd_000100000	DECOD2	D	Overwrite register "D (0-511)" with decoded D[1:0] repeated 8 times.
000011_zcr_1_cccc_ddddddddd_000100001	DECOD3	D	Overwrite register "D (0-511)" with decoded D[2:0] repeated 4 times.
000011_zcr_1_cccc_ddddddddd_000100010	DECOD4	D	Overwrite register "D (0-511)" with decoded D[3:0] repeated 2 times.
000011_zcr_1_cccc_ddddddddd_000100011	DECOD5	D	Overwrite register "D (0-511)" with decoded D[4:0] repeated 1 time.

Machine Code	Mnemonic	Operand	Operation
000011_zcr_1_cccc_ddddddddd_000100100	BLMASK	D	Overwrite register “D (0-511)” with a bit length mask specified by D[5:0].
000011_zcr_1_cccc_ddddddddd_000100101	NOT	D	Overwrite register “D (0-511)” with the bitwise inverted register “D (0-511)”.
000011_zcr_1_cccc_ddddddddd_000100110	ONECNT	D	Overwrite register “D (0-511)” with the count of ones in register D.
000011_zcr_1_cccc_ddddddddd_000100111	ZERCNT	D	Overwrite register “D (0-511)” with the count of zeros in register D.
000011_zcr_1_cccc_ddddddddd_000101000	INCPAT	D	Overwrite register “D (0-511)” with the next bit pattern that keeps the number of ones and zeros the same in register D.
000011_zcr_1_cccc_ddddddddd_000101001	DECPAT	D	Overwrite register “D (0-511)” with the previous bit pattern that keeps the number of ones and zeros the same in register D.
000011_zcr_1_cccc_ddddddddd_000101010	BINGRY	D	Overwrite the binary pattern in register “D (0-511)” with its gray code pattern.
000011_zcr_1_cccc_ddddddddd_000101011	GRYBIN	D	Overwrite the grey code pattern in register “D (0-511)” with its binary pattern.
000011_zcr_1_cccc_ddddddddd_000101100	MERGEW	D	Merge the high word and the low word of register “D (0-511)” into each other and overwrite register D with the new value. Bits of the low word occupy bit spaces 0, 2, 4, etc. Bits of the high word occupy bit spaces 1, 3, 5, etc. (Interleave)
000011_zcr_1_cccc_ddddddddd_000101101	SPLITW	D	Split the bits of register “D (0-511)” into a high word and low word and overwrite register D with the new value. Bits of the low word come from bit spaces 0, 2, 4, etc. Bits of the high word come from bit spaces 1, 3, 5, etc. (De-interleave)
000011_zcr_1_cccc_ddddddddd_000101110	SEUSSF	D	Overwrite register “D (0-511)” with a pseudo random bit pattern seeded from the value in register D. After 32 forward iterations, the original bit pattern is returned.
000011_zcr_1_cccc_ddddddddd_000101111	SEUSSR	D	Overwrite register “D (0-511)” with a pseudo random bit pattern seeded from the value in register D. After 32 reversed iterations, the original bit pattern is returned.
000011_zcr_1_cccc_ddddddddd_1000bbbbb	ISOB	D.b	Isolate bit “b (0-31)” of register “D (0-511).”
000011_zcr_1_cccc_ddddddddd_1001bbbbb	NOTB	D.b	Invert bit “b (0-31)” of register “D (0-511).”
000011_zcr_1_cccc_ddddddddd_1010bbbbb	CLRB	D.b	Clear bit “b (0-31)” of register “D (0-511).”
000011_zcr_1_cccc_ddddddddd_1011bbbbb	SETB	D.b	Set bit “b (0-31)” of register “D (0-511).”
000011_zcr_1_cccc_ddddddddd_1100bbbbb	SETBC	D.b	Set bit “b (0-31)” of register “D (0-511) to C.”
000011_zcr_1_cccc_ddddddddd_1101bbbbb	SETBNC	D.b	Set bit “b (0-31)” of register “D (0-511) to NC.”
000011_zcr_1_cccc_ddddddddd_1110bbbbb	SETBZ	D.b	Set bit “b (0-31)” of register “D (0-511) to Z.”
000011_zcr_1_cccc_ddddddddd_1111bbbbb	SETBNZ	D.b	Set bit “b (0-31)” of register “D (0-511) to NZ.”

Table 11: Extended Miscellaneous Flag Manipulation Instructions

Machine Code	Mnemonic	Operand	Operation
000011_zcr_1_cccc_ddddddddd_000001001	SWAPZC	D	Swap the Z and C flags with D[1:0] through WZ and WC effects.
000011_zcr_1_cccc_ddddddddd_000001010	PUSHZC	D	Push the Z and C flags into D[1:0] and pop D[31:30] into Z and C through WZ and WC.
000011_zcr_1_cccc_ddddddddd_000001011	POPZC	D	Pop D[1:0] into the Z and C flags and push D[31:30] into Z and C through WZ and WC.
000011_zcn_1_cccc_nnnnnnnnn_010100001	SETZC	D/#n	Set the Z and C flags with D[1:0] through WZ and WC effects.

Table 12: Extended Miscellaneous Flow Control Instructions

Machine Code	Mnemonic	Operand	Operation
000011_zcn_1_cccc_nnnnnnnnn_0100iiiii	REP	D/#n, #i	Repeat the following "i (0-31)" instructions the value in register "D(0-511)" or "n(0-511)" times.
000011_zcn_1_cccc_nnnnnnnnn_010100000	NOPX	D/#n	Repeat the NOP instruction the value in register "D(0-511)" or "n(0-511)" times.

Table 13: Miscellaneous Legacy Instructions

Machine Code	Mnemonic	Operand	Operation
000110_zcr_i_cccc_ddddddddd_sssssssss	ENC	D,S	Store encoded S in D.
000111_zcr_i_cccc_ddddddddd_sssssssss	JMPRET	D,S	See P8X32A – No instruction change.
001000_zcr_i_cccc_ddddddddd_sssssssss	ROR	D,S	See P8X32A – No instruction change.
001001_zcr_i_cccc_ddddddddd_sssssssss	ROL	D,S	See P8X32A – No instruction change.
001010_zcr_i_cccc_ddddddddd_sssssssss	SHR	D,S	See P8X32A – No instruction change.
001011_zcr_i_cccc_ddddddddd_sssssssss	SHL	D,S	See P8X32A – No instruction change.
001100_zcr_i_cccc_ddddddddd_sssssssss	RCR	D,S	See P8X32A – No instruction change.
001101_zcr_i_cccc_ddddddddd_sssssssss	RCL	D,S	See P8X32A – No instruction change.
001110_zcr_i_cccc_ddddddddd_sssssssss	SAR	D,S	See P8X32A – No instruction change.
001111_zcr_i_cccc_ddddddddd_sssssssss	REV	D,S	See P8X32A – No instruction change.
010000_zcr_i_cccc_ddddddddd_sssssssss	MINS	D,S	See P8X32A – No instruction change.
010001_zcr_i_cccc_ddddddddd_sssssssss	MAXS	D,S	See P8X32A – No instruction change.
010010_zcr_i_cccc_ddddddddd_sssssssss	MIN	D,S	See P8X32A – No instruction change.
010011_zcr_i_cccc_ddddddddd_sssssssss	MAX	D,S	See P8X32A – No instruction change.
010100_zcr_i_cccc_ddddddddd_sssssssss	MOVS	D,S	See P8X32A – No instruction change.
010101_zcr_i_cccc_ddddddddd_sssssssss	MOVD	D,S	See P8X32A – No instruction change.
010110_zcr_i_cccc_ddddddddd_sssssssss	MOVI	D,S	See P8X32A – No instruction change.
010111_zcr_i_cccc_ddddddddd_sssssssss	JMPRETD	D,S	See P8X32A – No instruction change. Do not flush pipeline before jump – must be executed two instructions before intended jump space.
011000_zcr_i_cccc_ddddddddd_sssssssss	AND	D,S	See P8X32A – No instruction change.
011001_zcr_i_cccc_ddddddddd_sssssssss	ANDN	D,S	See P8X32A – No instruction change.
011010_zcr_i_cccc_ddddddddd_sssssssss	OR	D,S	See P8X32A – No instruction change.
011011_zcr_i_cccc_ddddddddd_sssssssss	XOR	D,S	See P8X32A – No instruction change.
011100_zcr_i_cccc_ddddddddd_sssssssss	MUXC	D,S	See P8X32A – No instruction change.
011101_zcr_i_cccc_ddddddddd_sssssssss	MUXNC	D,S	See P8X32A – No instruction change.
011110_zcr_i_cccc_ddddddddd_sssssssss	MUXZ	D,S	See P8X32A – No instruction change.
011111_zcr_i_cccc_ddddddddd_sssssssss	MUXNZ	D,S	See P8X32A – No instruction change.
100000_zcr_i_cccc_ddddddddd_sssssssss	ADD	D,S	See P8X32A – No instruction change.
100001_zcr_i_cccc_ddddddddd_sssssssss	SUB	D,S	See P8X32A – No instruction change.

Machine Code	Mnemonic	Operand	Operation
100010_zcr_i_cccc_ddddddddd_sssssssss	ADDABS	D,S	See P8X32A – No instruction change.
100011_zcr_i_cccc_ddddddddd_sssssssss	SUBABS	D,S	See P8X32A – No instruction change.
100100_zcr_i_cccc_ddddddddd_sssssssss	SUMC	D,S	See P8X32A – No instruction change.
100101_zcr_i_cccc_ddddddddd_sssssssss	SUMNC	D,S	See P8X32A – No instruction change.
100110_zcr_i_cccc_ddddddddd_sssssssss	SUMZ	D,S	See P8X32A – No instruction change.
100111_zcr_i_cccc_ddddddddd_sssssssss	SUMNZ	D,S	See P8X32A – No instruction change.
101000_zcr_i_cccc_ddddddddd_sssssssss	MOV	D,S	See P8X32A – No instruction change.
101001_zcr_i_cccc_ddddddddd_sssssssss	NEG	D,S	See P8X32A – No instruction change.
101010_zcr_i_cccc_ddddddddd_sssssssss	ABS	D,S	See P8X32A – No instruction change.
101011_zcr_i_cccc_ddddddddd_sssssssss	ABSNEG	D,S	See P8X32A – No instruction change.
101100_zcr_i_cccc_ddddddddd_sssssssss	NEGC	D,S	See P8X32A – No instruction change.
101101_zcr_i_cccc_ddddddddd_sssssssss	NEGNC	D,S	See P8X32A – No instruction change.
101110_zcr_i_cccc_ddddddddd_sssssssss	NEGZ	D,S	See P8X32A – No instruction change.
101111_zcr_i_cccc_ddddddddd_sssssssss	NEGNZ	D,S	See P8X32A – No instruction change.
110000_zcr_i_cccc_ddddddddd_sssssssss	CMPS	D,S	See P8X32A – No instruction change.
110001_zcr_i_cccc_ddddddddd_sssssssss	CMPSX	D,S	See P8X32A – No instruction change.
110010_zcr_i_cccc_ddddddddd_sssssssss	ADDX	D,S	See P8X32A – No instruction change.
110011_zcr_i_cccc_ddddddddd_sssssssss	SUBX	D,S	See P8X32A – No instruction change.
110100_zcr_i_cccc_ddddddddd_sssssssss	ADDS	D,S	See P8X32A – No instruction change.
110101_zcr_i_cccc_ddddddddd_sssssssss	SUBS	D,S	See P8X32A – No instruction change.
110110_zcr_i_cccc_ddddddddd_sssssssss	ADDSX	D,S	See P8X32A – No instruction change.
110111_zcr_i_cccc_ddddddddd_sssssssss	SUBSX	D,S	See P8X32A – No instruction change.
111000_zcr_i_cccc_ddddddddd_sssssssss	SUBR	D,S	Subtract D from S and store in D.
111001_zcr_i_cccc_ddddddddd_sssssssss	CMPSUB	D,S	See P8X32A – No instruction change.
111010_zcr_i_cccc_ddddddddd_sssssssss	INCMOD	D,S	Increment D between 0 and S. Wraps around to zero when above S.
111011_zcr_i_cccc_ddddddddd_sssssssss	DECMOD	D,S	Decrement D between S and 0. Wraps around to S when below 0.
111100_00r_i_cccc_ddddddddd_sssssssss	IJZ	D,S	Increment D and jump to S if D is zero.
111100_01r_i_cccc_ddddddddd_sssssssss	IJZD	D,S	Increment D and jump to S if D is zero. Do not flush pipeline before jump – must be executed two instructions before intended jump space.
111100_10r_i_cccc_ddddddddd_sssssssss	IJNZ	D,S	Increment D and jump to S if D is not zero.
111100_11r_i_cccc_ddddddddd_sssssssss	IJNZD	D,S	Increment D and jump to S if D is not zero. Do not flush pipeline before jump – must be executed two instructions before intended jump space.
111101_00r_i_cccc_ddddddddd_sssssssss	DJZ	D,S	Decrement D and jump to S if D is zero.
111101_01r_i_cccc_ddddddddd_sssssssss	DJZD	D,S	Decrement D and jump to S if D is zero. Do not flush pipeline before jump – must be executed two instructions before intended jump space.

Machine Code	Mnemonic	Operand	Operation
111101_10r_i_cccc_ddddddddd_sssssssss	DJNZ	D,S	Decrement D and jump to S if D is not zero.
111101_11r_i_cccc_ddddddddd_sssssssss	DJNZD	D,S	Decrement D and jump to S if D is not zero. Do not flush pipeline before jump – must be executed two instructions before intended jump space.
111110_000_i_cccc_ddddddddd_sssssssss	TJZ	D,S	See P8X32A – No instruction change.
111110_010_i_cccc_ddddddddd_sssssssss	TJZD	D,S	See P8X32A – No instruction change. Do not flush pipeline before jump – must be executed two instructions before intended jump space.
111110_100_i_cccc_ddddddddd_sssssssss	TJNZ	D,S	See P8X32A – No instruction change.
111110_110_i_cccc_ddddddddd_sssssssss	TJNZD	D,S	See P8X32A – No instruction change. Do not flush pipeline before jump – must be executed two instructions before intended jump space.
111110_001_i_cccc_ddddddddd_sssssssss	SETINDA	D,S	Setup indirection register address A bottom range and top range where D is the top of the range and S is the bottom range. The indirection register will allow access to cog registers in this range.
111110_011_i_cccc_ddddddddd_sssssssss	SETINDB	D,S	Setup indirection register address B bottom range and top range where D is the top of the range and S is the bottom range. The indirection register will allow access to cog registers in this range.
111110_111_i_cccc_ddddddddd_sssssssss	WAITVID	D,S	Wait to pass pixels to the video generator.
111111_0cr_i_cccc_ddddddddd_sssssssss	WAITCNT	D,S	Wait for the CNT[31:0] register to equal D and then add S to D and store in D. If WC is specified then wait for CNT[63:32] to equal D.
111111_1c0_i_cccc_ddddddddd_sssssssss	WAITPEQ	D,S	See P8X32A – No instruction change.
111111_1c1_i_cccc_ddddddddd_sssssssss	WAITPNE	D,S	See P8X32A – No instruction change.

Register Map

Each cog has 10 memory mapped registers that allow control over I/O pins and indirection. The OUT and IN registers have now been combined to form the PIN registers. The IND registers allow indirect register access to avoid self-modifying code. All other REGs are free.

Table 14: Register Map Setup

Register	Location	Operation
INDA	0x1F6	When read or written writes to the cog memory address set my SETINDA. After being accessed auto increments. Condition codes are not allowed to be used with INDA register access.
INDB	0x1F7	When read or written writes to the cog memory address set my SETINDB. After being accessed auto increments. Condition codes are not allowed to be used with INDB register access.
PINA	0x1F8	When written changes the state of the I/O pin attached to port A. When read, returns the state of the I/O port attached to PINA.
PINB	0x1F9	When written changes the state of the I/O pin attached to port B. When read, returns the state of the I/O port attached to PINB.
PINC	0x1FA	When written changes the state of the I/O pin attached to port B. When read, returns the state of the I/O port attached to PINC.
PIND	0x1FB	When written changes the state of the I/O pin attached to port B. When read, returns the state of the I/O port attached to PIND.
DIRA	0x1FC	Enables or disables the output functionally of PORTA. Input reading is never disabled.
DIRB	0x1FD	Enables or disables the output functionally of PORTB. Input reading is never disabled.
DIRC	0x1FE	Enables or disables the output functionally of PORTC. Input reading is never disabled.
DIRD	0x1FF	Enables or disables the output functionally of PORTD. Input reading is never disabled.

Ports

There are now 4 I/O ports built into the system – 3 are physical 32-bit I/O ports and 1 is an internal 32-bit I/O port. The I/O pins connected to each port can be configured separately.

Table 15: Port Access Instructions

Machine Code	Mnemonic	Operand	Operation
000011_zcn_1_cccc_ddnnddddd_011100100	SETPORA	D/#n	Assign PORTA to physical I/O ports (0-2) or internal I/O port 3 given register "D (0-511)" or number "n (0-3)".
000011_zcn_1_cccc_ddnnddddd_011100101	SETPORB	D/#n	Assign PORTB to physical I/O ports (0-2) or internal I/O port 3 given register "D (0-511)" or number "n (0-3)".
000011_zcn_1_cccc_ddnnddddd_011100110	SETPORC	D/#n	Assign PORTC to physical I/O ports (0-2) or internal I/O port 3 given register "D (0-511)" or number "n (0-3)".
000011_zcn_1_cccc_ddnnddddd_011100111	SETPORD	D/#n	Assign PORTC to physical I/O ports (0-2) or internal I/O port 3 given register "D (0-511)" or number "n (0-3)".

Table 16: Pin State Access Instructions

Machine Code	Mnemonic	Operand	Operation
000011_zcn_1_cccc_ddnnnnnnn_011010110	GETP	D/#n	Get pin number given by register "D (0-511)" or "n (0-127)"into !Z or C flags.
000011_zcn_1_cccc_ddnnnnnnn_011010111	GETPN	D/#n	Get pin number given by register "D (0-511)" or "n (0-127)"into Z or !C flags.
000011_zcn_1_cccc_ddnnnnnnn_011011000	OFFP	D/#n	Toggle pin number given by register "D (0-511)" or "n (0-127)" off or on. DIR
000011_zcn_1_cccc_ddnnnnnnn_011011001	NOTP	D/#n	Invert pin number given by the value in register "D (0-511)" or "n (0-127)". OUT
000011_zcn_1_cccc_ddnnnnnnn_011011010	CLRP	D/#n	Clear pin number given by the value in register "D (0-511)" or "n (0-127)". OUT
000011_zcn_1_cccc_ddnnnnnnn_011011011	SETP	D/#n	Set pin number given by the value in register "D (0-511)" or "n (0-127)". OUT
000011_zcn_1_cccc_ddnnnnnnn_011011100	SETPC	D/#n	Set pin number given by the value in register "D (0-511)" or "n (0-127)" to C.
000011_zcn_1_cccc_ddnnnnnnn_011011101	SETPNC	D/#n	Set pin number given by the value in register "D (0-511)" or "n (0-127)" to !C.
000011_zcn_1_cccc_ddnnnnnnn_011011110	SETPZ	D/#n	Set pin number given by the value in register "D (0-511)" or "n (0-127)" to !Z.
000011_zcn_1_cccc_ddnnnnnnn_011011111	SETPNZ	D/#n	Set pin number given by the value in register "D (0-511)" or "n (0-127)" Z.

External RAM

Each cog now features the ability, with the help of the I/O pins, to quickly stream parallel data in or out of the I/O pins aligned to a clock source. Data is streamed to/from the CLUT or WRQUAD overlay. From there it can be quickly feed to the video generator or to the internal HUB RAM. XFR feeds data 16 Bits or 32 Bits at a time at the system clock speed.

Table 17: External RAM Instruction

Machine Code	Mnemonic	Operand	Operation
000011_zcn_1_cccc_dddnnnnnn_011101001	SETXFR	D/#n	Setup the direction of the data stream, the source and destination of the data stream, and the size of the data stream given D or "n (0-63)".

Chip-To-Chip Communication

Each cog now also features high-speed serial transfer and receive hardware for chip-to-chip communication. The hardware requires three I/O pins (SO, SI, CLK).

Table 18: Chip-To-Chip Communication Instructions

Machine Code	Mnemonic	Operand	Operation
000011 zc0 1 cccc ddddddddd 000001000	SNDSER	D	Sends a long (D) out of the special chip-to-chip serial port. Blocks until the long is sent. Use C flag to avoid blocking.
000011 zc1 1 cccc ddddddddd 000001000	RCVSER	D	Receives a long (D) in from the special chip-to-chip serial port. Blocks until the long is received. Use C flag to avoid blocking.
000011 zcn 1 cccc ddddddddd 011101010	SETSER	D/#n	Sets up the serial port I/O pins to use for SO, SI, and CLK given D or "n (0-63)".

Cog Memory Remapping

Cogs now have the ability to remap their internal memory to help facilitate context switching between register banks. Instead of having to save a bunch of internal register to switch running programs all references to a set of register can be changed instantaneously.

Table 19: Cog Memory Remapping Instruction

Machine Code	Mnemonic	Operand	Operation
000011_zcn_1_cccc_dnnnnnnnn_011100001	SETMAP	D/#n	Remap one cog register space to another cog register space given D or n.

Cog-To-Cog Communication

Cogs now have the ability to communicate directly to each other using the internal I/O Port D, which connects each cog to every other cog.

Table 20: Cog-To-Cog Communication Instruction

Machine Code	Mnemonic	Operand	Operation
000011_zcn_1_cccc_nnnnnnnnn_011101000	SETXCH	D/#n	Reconfigure Port D I/O masks given D or n to select which cogs to listen to.

Pin Modes

Each I/O pin is now capable of setting itself into many different modes to more easily interface with the analog world. By default, each I/O starts up in the basic robust digital I/O state. However, once configured the I/O pin can be used for external RAM memory transfer, or as an ADC, or as a DAC, or a Schmitt trigger, or a comparator, etc.

Table 21: Pin Mode Access Instructions

Machine Code	Mnemonic	Operand	Operation
000011_zcn_1_cccc_ddnndddddd_011100011	SETPORT	D/#n	Assign which port the CFGPINS instruction will configure given register "D (0-511)" or number "n (0-3)".
111110_101_i_cccc_ddddddddd_sssssssss	CFGPINS	D,S	Setup pins masked by register "D (0-511)" to register "S (0-511)". The pin configuration modes are below.

NOTE: PinA is the pin being set. PinB is its neighbor (All I/O pins have a cross coupled neighbor). Input is the Boolean statement for what the pin returns when read. Output is the statement for what the pins outputs when it is an output (Some modes output their input to make feedback relaxation oscillators, etc). Each pin's high and low drivers can be configured to work in many different modes. Pins can also re-clock data sent to them locally to remove jitter in data. Every pin is setup by a 13-bit configuration value.

Figure 2: Pin Modes

Code	Mode	Input	PinA Output	PinB	Compare
0000_CIOHHHLLL	General I/O	PinA Logic	OUT	-	-
0001_CIOHHHLLL	DIR=0 Float	PinA Logic	Input	-	-
0010_CIOHHHLLL		PinB Logic	Input	-	-
0011_CIOHHHLLL	C | OUT/IN; DIR=1 Drive	PinB Logic	Input	1MΩ PinA	-
0100_CIOHHHLLL	0 | Live; HHH LLL | Drive	PinA Schmitt	OUT	-	-
0101_CIOHHHLLL	1 | Clocked	PinA Schmitt	Input	-	-
0110_CIOHHHLLL	000 | Fast	PinB Schmitt	Input	-	-
0111_CIOHHHLLL	I | IN; 001 | Slow	PinB Schmitt	Input	1MΩ PinA	-
1000_CIOHHHLLL	0 | True; 010 | 1500Ω	PinA > VIO/2	OUT	-	Fast
1001_CIOHHHLLL	1 | Inverted; 011 | 10kΩ	PinA > VIO/2	Input	-	Fast
1010_CIOHHHLLL	100 | 100kΩ	PinB > VIO/2	Input	-	Fast
1011_CIOHHHLLL	O | Output; 101 | 100µA	PinB > VIO/2	Input	1MΩ PinA	Fast
1100_CIOHHHLLL	0 | True; 110 | 10µA	PinA > PinB	OUT	-	Precise
1101_CIOHHHLLL	1 | Inverted; 111 | Float	PinA > PinB	Input	-	Precise
1110_CIOHHHLLL		PinA > PinB	Input	1MΩ PinA	Precise
1111_0LLLLLLL	Compare Level	PinA > VIO/256*L	-	-	Precise
1111_1000xxxxx	ADC Diff, 100kΩ	PinA > VIO/2 10kΩ	100kΩ, !IN	10kΩ VIO/2	Fast
1111_10010xxxx	ADC Precise, DIR/OUT=Cal	ADC	7MΩ	-	Fast
1111_10011xxxx	ADC Fast, DIR/OUT=Cal	ADC	400kΩ	-	Fast
1111_101VxxCCC	DAC 75Ω, V=Video, C=Cog	1	75Ω	-	-
1111_110HHHLLL	SDRAM Data I/O	PinA Logic	Fast, OUT	-	-
1111_111HHHLLL	SDRAM Clock Out	1	Fast, OUT=1	-	-

Video Generator

Each cog has a video generator capable of generating composite, component, s-video, and VGA video. The video generator is fed pixel data through the waitvid instruction and uses the pixel data to look up colors to output from the CLUT. The video generator understands R.G.B.A.X color grouping and can handle RGB565/555/444/etc formatted data.

Table 22: Video Generator Access Instructions

Machine Code	Mnemonic	Operand	Operation
000011_zcn_1_cccc_dddnnnnnn_011101011	SETVID	D/#n	Setup the video generator according to D or n to output video from the CLUT.
000011_zcn_1_cccc_nnnnnnnnn_011101100	SETVIDM	D/#n	Setup the video generator color matrix transform term M according to D or n.
000011_zcn 1_cccc_nnnnnnnnn_011101101	SETVIDY	D/#n	Setup the video generator color matrix transform term Y according to D or n.
000011_zcn_1_cccc _nnnnnnnn_011101110	SETVIDI	D/#n	Setup the video generator color matrix transform term I according to D or n.
000011_zcn_1_cccc_nnnnnnnnn_011101111	SETVIDQ	D/#n	Setup the video generator color matrix transform term Q according to D or n.

DAC Hardware

Each cog has four DACs capable of SIN/COS wave output, saw tooth wave output, triangle wave output, and square wave output. Additionally, the video generator, when operational, will use the four DACs to produce video output. Please refer to the information below.

- CFGDAC – 00 = 9-bit level with 9-bit dither.
- CFGDAC – 01 = 9-bit level from counter with 9-bit dither from counter.
  - DAC0 = CTRASIN, DAC1 = CTRACOS, DAC2 = CTRBSIN, DAC3 = CTRBCOS
- CFGDAC – 10 = 9-bit level from counter with 9-bit dither from counter.
  - DAC0/2 = CTRASIN + CTRBSIN, DAC1.3 = CTRACOS + CTRBCOS
- CFGDAC – 11 = Video generator controlled.
  - DAC0 = SYNC, DAC1 = Q/B, DAC2 = I/G, DAC3 = Y/R

Table 23: DAC Hardware Access Instructions

Machine Code	Mnemonic	Operand	Operation
000011_zcn_1_cccc_ddddddnn_011001100	CFGDAC0	D/#n	Configure DAC0 to D or n. See above.
000011_zcn_1_cccc_ddddddnn_011001101	CFGDAC1	D/#n	Configure DAC0 to D or n. See above.
000011_zcn_1_cccc_ddddddnn_011001110	CFGDAC2	D/#n	Configure DAC0 to D or n. See above.
000011_zcn_1_cccc_ddddddnn_011001111	CFGDAC3	D/#n	Configure DAC0 to D or n. See above.
000011_zcn_1_cccc_nnnnnnnnn_011010000	SETDAC0	D/#n	Set DAC0 to top 18 bits of D/n.
000011_zcn_1_cccc_nnnnnnnnn_011010001	SETDAC1	D/#n	Set DAC1 to top 18 bits of D/n.
000011_zcn_1_cccc_nnnnnnnnn_011010010	SETDAC2	D/#n	Set DAC2 to top 18 bits of D/n
000011_zcn_1_cccc_nnnnnnnnn_011010011	SETDAC3	D/#n	Set DAC3 to top 18 bits of D/n.
000011_zcn_1_cccc_dnnnnnnnn_011010100	CFGDACS	D/#n	Configure DACs to D or n. See above.
000011_zcn_1_cccc_nnnnnnnnn_011010101	SETDACS	D/#n	Set DACs to top 18 bits of D/n.

Texture Mapping

Each cog has texture mapping hardware to assist the video generator with displaying textures and performing color blending on screen.

Table 24: Texture Mapping Instructions

Machine Code	Mnemonic	Operand	Operation
000011_zcr_1_cccc_ddddddddd_000010100	GETPIX	D	Store texture pointer address in D.
000011_zcn_1_cccc_nnnnnnnnn_010111000	SETPIX	D/#n	Set texture size and address to D/n.
000011_zcn_1_cccc_nnnnnnnnn_010111001	SETPIXU	D/#n	Set texture pointer x address to D/n.
000011_zcn_1_cccc_nnnnnnnnn_010111010	SETPIXV	D/#n	Set texture pointer y address to D/n.
000011_zcn_1_cccc_nnnnnnnnn_010111011	SETPIXZ	D/#n	Set texture pointer z address to D/n.
000011_zcn_1_cccc_nnnnnnnnn_010111100	SETPIXR	D/#n	Set texture pointer R blending to D/n.
000011_zcn_1_cccc_nnnnnnnnn_010111101	SETPIXG	D/#n	Set texture pointer G blending to D/n.
000011_zcn_1_cccc_nnnnnnnnn_010111110	SETPIXB	D/#n	Set texture pointer B blending to D/n.
000011_zcn_1_cccc_nnnnnnnnn_010111111	SETPIXA	D/#n	Set texture pointer A blending to D/n.

Counter Modules

Each cog has two counter modules – CTRA and CTRB. Each counter module has a FRQ, PHS, SIN, and COS register. The counter modules control the SIN and COS registers to track the phase and power of a signal. The FRQ and PHS registers work the same. Each counter module also has logic modes, which allow it to accumulate given different logic equations involving a selected pin A and pin B – see P8X32A. The counter modes now also feature quadrature encoder accumulation and automatic PWM generation.

Table 25: Counter Hardware Access Instructions

Machine Code	Mnemonic	Operand	Operation
000011_zcr_1_cccc_ddddddddd_000111000	GETPHSA	D	Store PHSA in D.
000011_zcr_1_cccc_ddddddddd_000111001	GETPHZA	D	Store PHSA in D and zero PHSA.
000011_zcr_1_cccc_ddddddddd_000111010	GETCOSA	D	Store COSA in D.
000011_zcr_1_cccc_ddddddddd_000111011	GETSINA	D	Store SINA in D.
000011_zcr_1_cccc_ddddddddd_000111100	GETPHSB	D	Store PHSB in D.
000011_zcr_1_cccc_ddddddddd_000111101	GETPHZB	D	Store PHSB in D and zero PHSB.
000011_zcr_1_cccc_ddddddddd_000111110	GETCOSB	D	Store COSB in D.
000011_zcr_1_cccc_ddddddddd_000111111	GETSINB	D	Store SINB in D.
000011_zcn_1_cccc_nnnnnnnnn_011110000	SETCTRA	D/#n	Set CTRA mode to D/n.
000011_zcn_1_cccc_nnnnnnnnn_011110001	SETWAVA	D/#n	Set CTRA wave mode to D/n.
000011_zcn_1_cccc_nnnnnnnnn_011110010	SETFRQA	D/#n	Set FRQA to D/n.
000011_zcn_1_cccc_nnnnnnnnn_011110011	SETPHSA	D/#n	Set PSHA to D/n.
000011_zcn_1_cccc_nnnnnnnnn_011110100	ADDPHSA	D/#n	Add D/n to PSHA.
000011_zcn_1_cccc_nnnnnnnnn_011110101	SUBPHSA	D/#n	Subtract D/n from PSHA.
000011_zcn_1_cccc_nnnnnnnnn_011110110	SYNCTRA		Wait for PHSA to overflow.
000011_zcn_1_cccc_nnnnnnnnn_011110111	CAPCTRA		Remove current sum from PHSA.
000011_zcn_1_cccc _nnnnnnnnn_011111000	SETCTRB	D/#n	Set CTRB mode to D/n.
000011_zcn_1_cccc_nnnnnnnnn_011111001	SETWAVB	D/#n	Set CTRB wave mode to D/n.
000011_zcn_1_cccc_nnnnnnnnn_011111010	SETFRQB	D/#n	Set FRQB to D/n.
000011_zcn_1_cccc_nnnnnnnnn_011111011	SETPHSB	D/#n	Set PSHB to D/n.
000011_zcn_1_cccc_nnnnnnnnn_011111100	ADDPHSB	D/#n	Add D/n to PSHB.
000011_zcn_1_cccc_nnnnnnnnn_011111101	SUBPHSB	D/#n	Subtract D/n from PSHB.
000011_zcn_1_cccc_nnnnnnnnn_011111110	SYNCTRB		Wait for PHSB to overflow.
000011_zcn_1_cccc_nnnnnnnnn_011111111	CAPCTRB		Remove current sum from PHSB.

Effects and Condition Codes

Every assembly instruction can conditionally update the Z and/or C flag with WC and WZ effects. Additionally, the result can conditionally be written using the NR and WR flags. In addition, instructions can be conditionally executed given the Z and/or C flag—see P8X32A.

Revision History

Version 1.0: Original document.
Version 1.1: Corrected Machine Code for RDLONG, WRLONG, and RDLONGC, in Table 1: Hub Memory Access Instructions on page 2.