SENSORY®

# RSC-464
## Speech Recognition Processor

**Data Sheet**

## General Description

The RSC-464 is the newest member of Sensory's RSC-4x Family of microcontrollers with on-chip speech I/O capabilities. The RSC-464 has many features of the RSC-4128, but reduced in cost by integrating less memory. The RSC-464 is designed to bring high performance speech I/O features to cost sensitive embedded and consumer products. Based on an 8-bit microcontroller, the RSC-464 integrates speech-optimized digital and analog processing blocks into a single chip solution capable of accurate speech recognition; high quality, low data-rate compressed speech; and advanced music. Products can use one or all features in a single application.

The RSC-464 operates in tandem with the radically new FluentChip™ technology, offering the best speech recognition technologies in the industry. FluentChip™ includes Hidden Markov Model-Neural Net hybrid speech recognition. Accuracy in all kinds of noise is dramatically improved. New Speaker Verification technology is perfect for voice password security applications that must work in noisy environments. New high quality compressed speech technology reduces data rates by 5 times. New 8-voice MIDI-compatible music includes drum tracks, effectively increasing instruments beyond 8. Simultaneous music and speech round out the FluentChip™ technology.

FluentChip™ technology tools also support the revolutionary capability of creating speaker independent recognition sets by simply typing in the desired recognition vocabulary! A few keystrokes creates a recognition set in seconds without the wait or cost of recording sessions to train the recognizer, speeding time to sales.

The Audio Wakeup feature listens while the RSC-464 is in power down mode. When an audio event such as a clap or whistle occurs, Audio Wakeup will wakeup the RSC-464 for speech or application tasks. Audio Wakeup is perfect for battery applications that require continuous listening and long battery life.

The RSC-464 provides further on-chip integration of features. A complete speech I/O application can be built with as few additional parts as a clock crystal, speaker, microphone, and few resistors and capacitors.

Moreover, the RSC-464 provides an unprecedented level of cost effective system-on-chip (SOC) integration, enabling many applications that require DSP and/or audio processing. The RSC-464 may be used as a general-purpose mixed signal processor platform for custom algorithms, technologies and applications.

## Features

### *Full Range of FluentChip™ Capabilities*

- Noise-robust Speaker Independent and Speaker Dependent recognition
- Many languages now available for international use
- Speaker Verification – voice password biometric security
- Word Spotting and Continuous Listening recognition options
- High quality, 2.4-10.8 kbps speech synthesis & sound effects, with Sensory SX™ synthesis technology
- 8 voice MIDI-compatible music synthesis coincident with speech; drum track feature enables additional voices
- Voice Record & Playback (voice memo)
- Audio Wake Up from sleep with whistles or claps
- Touch Tone (DTMF) output

### *Integrated Single-Chip Solution*

- 8-bit microcontroller
- 64K bytes ROM
- 16 bit ADC, 10 bit DAC & PWM, and microphone pre-amplifier; PWM 30% louder than before!
- Independent, programmable Digital Filter engine
- 2.8 KBytes total RAM (262 bytes "user" application RAM)
- Five timers (3 GP, 1 Watchdog, 1 Multi Tasking)
- Twin-DMA, Vector Math accelerator, and Multiplier
- Built-in Analog Comparator Unit (4 inputs)
- On chip storage for SD, SV, templates
- 16 configurable I/O lines with 10 mA (typical) outputs
- Uses low cost 3.58MHz crystal (internal PLL)
- Low EMI design for FCC and CE requirements
- Fully nested interrupt structure with up to 8 sources
- Optional Real Time Clock

### *Long Battery Life*

- 2.4 – 3.6V operation
- 10mA (typical) operating current at 3V during
- 2 low power modes; 1 μA typical sleep current

### *Full Suite of Quick & Powerful Tools*

- Quick Text-to-SI (T2SI) text entry to build noise robust SI recognition sets – low cost & push-button – no recording!
- Quick Synthesis for push-button speech compression
- Integrated Development Environment, C Compiler, Debugger & In Circuit Emulator from Phyton, Inc.

# Table of Contents



# RSC-464 Overview

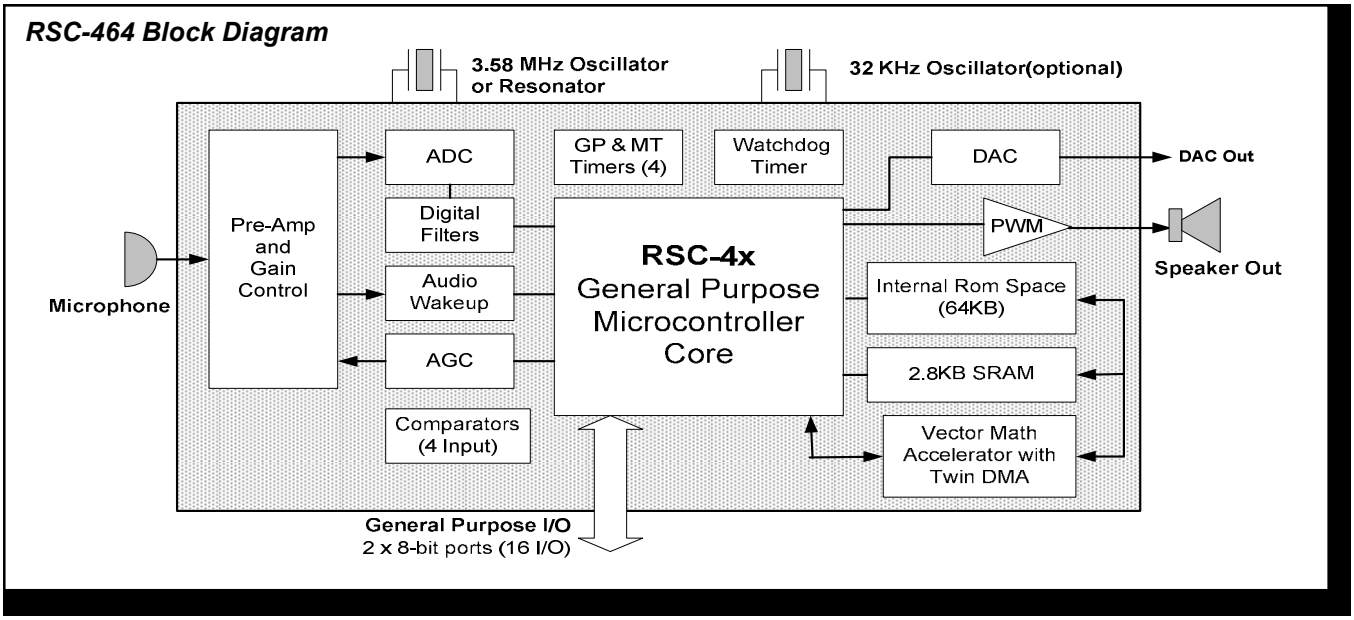
The RSC-464 is a member of the Interactive Speech™ line of products from Sensory. It features a high-performance 8-bit microcontroller with on-chip ADC, DAC, preamplifier, RAM, ROM, and optimized audio processing blocks. The RSC-464 is designed to bring a high degree of integration and versatility into low-cost, power-sensitive applications. Various functional units have been integrated onto the CPU core in order to reduce total system cost and increase system reliability.

The RSC-464 operates in tandem with FluentChip™ firmware, an ultra compact suite of recognition and synthesis technologies. This reduced software footprint enables, for example, products with 60 seconds of compressed speech, multiple speaker dependent and independent vocabularies, speaker verification, and all application code built into the RSC-464 as a single chip solution. Revolutionary Text-to-Speaker-Independent (T2SI) technology allows the creation of SI recognition sets by simply entering text.

The CPU core embedded in the RSC-464 is an 8-bit, variable-length-instruction microcontroller. The instruction set is similar to the 8051 microcontroller, and has a variety of addressing mode, *MOV* and 16 bit instructions. The RSC-464 processor avoids the limitations of dedicated A, B, and DPTR registers by having completely symmetrical sources and destinations for all instructions.

The RSC-464 provides a high level of on-chip features and special DSP engines, providing a very cost effective mixed signal platform for general-purpose applications and development of custom algorithms. The full suite of industry standard tools for easy product development makes the RSC-464 an ideal platform for consumer electronics.

***RSC-464 Block Diagram***



# Speech Technologies

## Speech Recognition

The RSC-464 is designed to operate in tandem with the FluentChip™ technology library, including speaker independent (SI), speaker dependent (SD), and speaker verification (SV) speech recognition. Combinations of these technologies may used to create applications that are rich in features. These are described below:

- ***Speaker Independent*** recognition requires no user training. The RSC-464 can recognize up to 15 commands in an active set (number of sets is limited only by internal ROM size). Text-to-SI (T2SI), based on a hybrid of Hidden Markov Modeling and Neural Net technologies, allows creation of accurate SI recognition sets in seconds. SI requires on-chip ROM.
- ***Speaker Dependent*** recognition allows the user to create names for products or customize recognition sets. SD is implemented with DTW (dynamic time warping) pattern matching technology. SD requires programmable memory to store the personalized speech templates(trained patterns) that may be on-chip SRAM, or off-chip serial EEPROM, Flash Memory, or SRAM. Up to 50 templates can be recognized in an active set (the number of unique sets is limited only by programmable memory capacity). The RSC-464 can store 1 SD templates in on-chip SRAM.
- ***Speaker Verification*** enables the RSC-464 to authenticate when a previously trained password is spoken by the target user. SV is also implemented with DTW technology. 1 SV template can be stored in on-chip SRAM, or more with external programmable memory such as delineated in SD above.
- ***Word Spotting*** enables the RSC-464 to spot a specific word surrounded by other speech within a phrase. This can be quite effective when the users response may vary (e.g. spotting “telephone” in the phrases “ummm telephone”, or “telephone call”). This option is available for SI and SD.
- ***Continuous Listening*** allows the chip to continuously listen for a specific word. This may be used as a trigger word to request a device to listen for a command. This option is available for SI and SD.

## Speech and Music Synthesis

The RSC-464 provides high-quality speech compression using Sensory SX™ technology. One may select various data rates from approximately 2.4 to 10.8Kbps to manage speech quality versus allotted memory. The highest data rates use 16KHz sample rates to provide high quality reproduction of high pitched voices. Speech and sound effects may also be compressed using 8-bit PCM (64Kbps) or 4-bit ADPCM (32Kbps) technologies.

The RSC-464 also provides eight-voice, wave table music synthesis which allows multiple, simultaneous instruments for harmonizing. The RSC-464 uses a MIDI-like system to generate music. One or more of the eight voices may be speech playback instead of music. One or more of the eight voices may be a drum track comprising multiple drums. In effect, drum tracks allow the number of simultaneous instruments to exceed 8.

Speech and Music data may be stored in on-chip ROM. Speech data may alternatively be stored in off-chip serial data ROM or serial data Flash for extended durations.

Easy to use tools allow the developer to record and compress their own voice talents and create with the push of a button, or to create their own MIDI scores and instruments.

## Record and Playback

The RSC-464 can perform speech record and playback (sometimes called “voice memo”) using either 8 bits (64Kbps) or 4 bits (32Kbps) per sample, depending on the quantity and quality of playback desired. The record and playback technology also optionally performs silence removal to reduce memory requirements.

External serial Flash or SRAM is required to store the compressed speech.

# RSC-464 Architecture

The RSC-464 is a highly integrated speech and analog I/O mixed signal processor that combines:

- 8-bit microcontroller with enhanced instructions and interrupt control, superior register architecture, independent Digital Filter engine and "L1" Vector Math Accelerator
- On-chip ROM and RAM (2.8 Kbytes).
- Input microphone preamp and 16 bit Analog-to-Digital Converter (ADC) for speech and audio/analog input
- 10 bit Digital-to-Analog Converter (DAC), and 10 bit Pulse Width Modulator (PWM) to directly drive a speaker or other analog device
- Low power Audio Wakeup from power down mode, when a selected audio event, such as clap or whistle, occurs

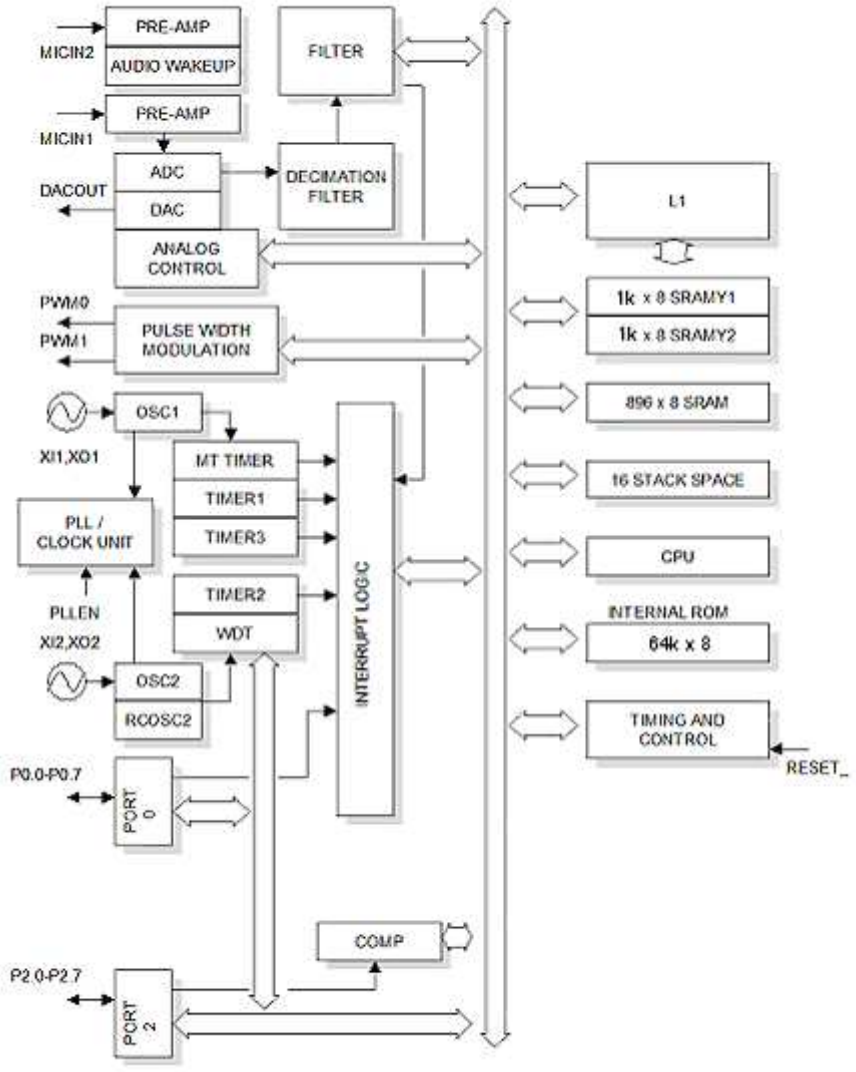
Two bi-directional ports provide 16 configurable, general-purpose I/O pins to communicate with or control external devices with a variety of source and sink currents. Up to 4 of these I/O may be used as programmable Analog Comparator inputs. 16 may be used as I/O wakeup.

The RSC-464 has a high frequency (14.32 MHz) clock as well as a low frequency (32,768 Hz) clock. The processor clock can be selected from either source, with a selectable divider value. The device performs speech recognition when running at 14.32 MHz.

OSC1 is a very low-cost 3.58 MHz crystal oscillator that is used by a 4X PLL to generate the 14.32MHz clock. The OSC2 oscillator provides the options of using an external crystal or its own internal RC devices (no external components required for the internal RC mode).

There are three programmable, general-purpose 8-bit counters / timers – Timers 1 and 3 are derived from OSC1, and Timer2 from OSC2. There is also a Watchdog timer that may be used to exit an undesired condition in program flow, and Multi-tasking timer to allow chip operations to share resources in parallel.

***RSC-464 Internal Block Diagram***




A single chip speech I/O solution may be created with the RSC-464. An external microphone passes an audio signal to the preamplifier and ADC to convert the incoming speech signal into digital data. Speech features are extracted using the Digital Filter engine. The microcontroller CPU processes these speech features using speech recognition algorithms in firmware, with the help of the "L1" Vector Accelerator and enhanced instruction set. The resulting speech recognition results may be used to control the consumer product application code, or

to output speech or audio in the form of a dialog with the user of the consumer product. If desired, the output speech or audio signal from the RSC-464 is generated by a DAC for external amplification into a speaker, or a PWM capable of directly driving a speaker at typical consumer product volumes. A typical product will require about $0.30 - $1.00 (in high volume) of additional components, in addition to the RSC-464.

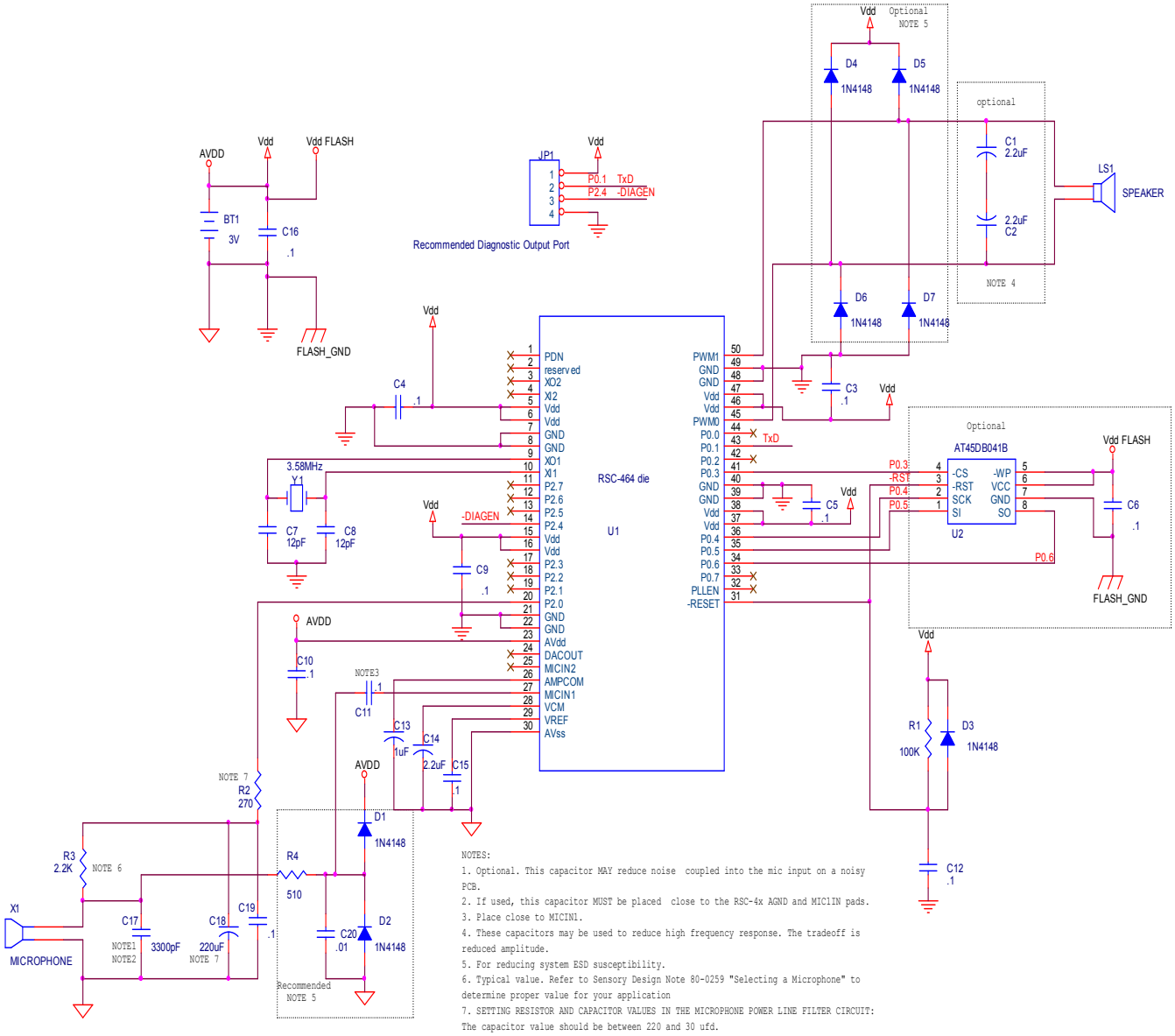
The RSC-464 also provides a very cost effective mixed signal platform for general-purpose applications and development of custom algorithms. A typical general-purpose application will require about $0.30 - $0.50 (in high volume) of additional components, in addition to the RSC-464.

# Reference Schematics

**Schematic 1-1:**

**RSC-464, Utilizing On-chip ROM and Optional External Serial Data Memory**




NOTES:

1. Optional. This capacitor MAY reduce noise coupled into the mic input on a noisy PCB.
2. If used, this capacitor MUST be placed close to the RSC-4x AGND and MIC1IN pads.
3. Place close to MICIN1.
4. These capacitors may be used to reduce high frequency response. The tradeoff is reduced amplitude.
5. For reducing system ESD susceptibility.
6. Typical value. Refer to Sensory Design Note 80-0259 "Selecting a Microphone" to determine proper value for your application
7. SETTING RESISTOR AND CAPACITOR VALUES IN THE MICROPHONE POWER LINE FILTER CIRCUIT:
   The capacitor value should be between 220 and 30 ufd.
   The resistor value should be between 300 and 2000 ohms.
   Within these ranges;
   Select the largest capacitor consistent with other constraints (cost, size,etc.)
   For this capacitor, select a resistor such that the RC time constant is ~60msec.

* Refer to Sensory Design Note 80-0073 "Speech Recognition Hardware Design" for information on proper microphone selection and housing design, PCB layout, as well as test and evaluation procedures to verify proper design and operation.
* Sensory offers a FREE Design Review Service. Detatils of this service are also included in 80-0073.

# Using the RSC-464

Creating applications using the RSC-464 requires the development of electronic circuitry, software code, and speech/music data files. Software code for the RSC-464 can be developed using a complete suite of RSC-464 development tools including In-Circuit Emulator, C Compiler, and "push button" tools for speech recognition and synthesis data files. Sensory provides free design reviews of customer applications to assist in the speech dialog and speech I/O design. Sensory also offers application development services. For more information about development tools and services, please contact Sensory.

When using the RSC-464 macro blocks such as the AFE, digital filters, L1, etc, for purposes other than as intended in the FluentChip™ technology modules, in applications that will also use FluentChip™ technologies, care must be taken to avoid conflicts that may cause adverse impact on functionality. Contact Sensory Technical Support for help in avoiding these conflicts.

## Instruction Set

The instruction set for the RSC-464 has 60 instructions comprising 13 move, 7 rotate, 11 branch, 22 arithmetic, and 7 miscellaneous instructions. All instructions are 3 bytes or fewer and no instruction requires more than 10 clock cycles (plus wait states) to execute. (see "Instruction Set Opcodes and Timing Details" for detailed descriptions)

## Flags

The "flags" register (register FF) has bits that are set/cleared by arithmetic/logical instructions, a trap enable bit set under program control, a read-only stack overflow bit cleared at power on and set by stack wrap around, and the Global Interrupt Enable bit:

| | | | |
|---|---|---|---|
| 0FFH | R/W | "flags" | |
| | Bit 7: | carry | |
| | Bit 6: | zero | (set = 1 when result of arith/log instruction is 0) |
| | Bit 5: | sign | (set = 1 when result of arith/log instruction has msb high) |
| | Bit 4: | trap | |
| | Bit 3: | stkoflo | (read-only, initialized to 0, set to 1 on stack overflow) |
| | Bit 3: | stkfull | (read-only, initialized to 0, set to 1 on stack full) |
| | Bit 1: | (unused) | |
| | Bit 0: | gie | (Global Interrupt Enable) |

NOTE: The "trap" bit must be left written as "0".

### *Flags Hold*

The "flagsHold" register (register CF) stores the "flags" value when an interrupt occurs. Unlike previous RSC chips, the RSC-464 processor has read/write access to "flagsHold" for multi-tasking purposes. Since the "flags" register is restored from the "flagsHold" register upon return from interrupt, the "stkoflo" and "stkfull" bits are omitted from the "flagsHold" register to prevent inadvertent clearing of these bits.

| | | |
|---|---|---|
| 0CFH | R/W | "flagsHold" |
| | Bit 7: | carry |
| | Bit 6: | zero |
| | Bit 5: | sign |
| | Bit 4: | trap |
| | Bit 3: | (unused – reads 0) |
| | Bit 2: | (unused – reads 0) |
| | Bit 1: | (unused – reads 0) |
| | Bit 0: | gie |

NOTE: The "trap" bit must be left written as "0".

See the discussion in "Interrupts" section relating to the value of "gie" stored in the "flagsHold" register when an interrupt occurs during execution of an instruction that clears the "gie" bit.

ERRATA NOTE: "Shadowing" of the "gie" flag bit in the flagsHold register is not disabled during an Interrupt Service Routine (ISR) as intended, due to design errata. Therefore, any operation that directly modifies the flags register "gie" bit (mov, logic operations, arithmetic operations, CLI, STI, etc directly on the flags register) will erroneously invoke a "shadowing" of the flags register "gie" bit in the flagsHold register.
To avoid this problem, simply store the flagsHold register in a temporary location immediately upon entering an ISR and restore the flagsHold register from that temporary location as the final instruction before exiting the ISR.

## Stack

There is a 16-level, 16-bit stack for saving the program counter for subroutine calls and interrupt requests. The stack pointer wraps around on overflow or underflow. When the stack read and write pointers indicate that stack overflow has occurred, the "stkoflo" bit in the "flags" register is set. Once set, this bit can only be cleared by a processor reset. The bit may be tested by software, but it performs no other function. When the stack read and write pointers indicate that stack is full, the "stkfull" bit in the "flags" register is set. This bit will be reset once the stack is not full.

### *Stack Pointers*

The 16-level stack has two 4-bit pointers, stack write and stack read. They are normally written by the processor upon execution of a "CALL" instruction or an interrupt.

The stack also has a 6-bit index register "stkNdx" (register F6) and an 8-bit data port register "stkData" (register F7) that are used to access the stack contents as bytes in a register file under program control. The contents of the stack location selected by the "stkNdx" register may be read or written by the processor via *MOV* instructions at the "stkData" register. The stack register index must be written first, then the stack data can be read.

The Stack read and write pointers (4 bits each) are also mapped to addresses accessible via the Stack Register Index.

Stack contents accessed by value in stack register index ("stkNdx", register F6)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 00H | Stack0 Lo | 08H | Stack4 Lo | 10H | Stack8 Lo | 18H | StackC Lo |
| 01H | Stack0 Hi | 09H | Stack4 Hi | 11H | Stack8 Hi | 19H | StackC Hi |
| 02H | Stack1 Lo | 0AH | Stack5 Lo | 12H | Stack9 Lo | 1AH | StackD Lo |
| 03H | Stack1 Hi | 0BH | Stack5 Hi | 13H | Stack9 Hi | 1BH | StackD Hi |
| 04H | Stack2 Lo | 0CH | Stack6 Lo | 14H | StackA Lo | 1CH | StackE Lo |
| 05H | Stack2 Hi | 0DH | Stack6 Hi | 15H | StackA Hi | 1DH | StackE Hi |
| 06H | Stack3 Lo | 0EH | Stack7 Lo | 16H | StackB Lo | 1EH | StackF Lo |
| 07H | Stack3 Hi | 0FH | Stack7 Hi | 17H | StackB Hi | 1FH | StacKF Hi |
| 20-2FH | (unused) | 30-3DH | (unused) | 3EH | StackWritePtr (4bits only) | 3FH | StackReadPtr (4bits only) |

## Register and User RAM

The RSC-464 has a physical register RAM space of 896 bytes. There is an additional RAM space of 64 bytes dedicated to Special Function Registers (SFRs), for a total register RAM space of 960 bytes. User RAM is assigned 262 bytes of this register RAM space, as detailed below.

Logical register space addressing is architecturally limited to 8 bits (256 bytes). Therefore a banking scheme is used to provide the total of 960 bytes of register RAM space. The lower 128 bytes and the top 64 bytes of addressing are used to directly address register RAM. The remaining 64 bytes (080H-0BFH) are banked to provide the remaining 768 bytes of register RAM space. This 768 bytes of register RAM is divided into 12 banks of 64 bytes each. The "bank" register (register FC) is combined with logical addressing to access these 12 banks. Here is a table illustrating the breakdown of register RAM space:

| | |
|---|---|
| 000H-07FH | unbanked register RAM |
| 080H-0BFH | banked register RAM |
| 0C0H-0FFH | unbanked register RAM (SFRs) |

Bits [4:0] of the "bank" register determine which physical bank of 64 bytes is logically mapped to addresses 080H-0BFH. When a logical address falls in the range of 080H-0BFH, the lower 6 bits of the logical address (64 byte address) are combined with the "bank" register bits used as the upper 5 bits of an 11-bit physical address. This physical address is used to address 768 bytes (12 banks) of physical bank RAM. (Note: 4 bits are required by the "bank" register to address 12 banks, but 5 bits are provided to allow for possible increases in the register RAM for future RSC family members.) Here is a table that illustrates this banking scheme:

Mapping of logical addresses 080H-0BFH ("bank" register FC is used)

| register FC [4:0] | Physical Bank RAM | | register FC [4:0] | Physical Bank RAM |
|---|---|---|---|---|
| 00H (Bank 0) | 00-3FH | | 08H (Bank 8) | 200-23FH |
| 01H (Bank 1) | 40-7FH | | 09H (Bank 9) | 240-27FH |
| 02H (Bank 2) | 80-BFH | | 0AH (Bank A) | 280-2BFH |
| 03H (Bank 3) | C0-FFH | | 0BH (Bank B) | 2C0-2FFH |
| 04H (Bank 4) | 100-13FH | | 0CH | --- (unimplemented) |
| 05H (Bank 5) | 140-17FH | | 0DH | --- (unimplemented) |
| 06H (Bank 6) | 180-1BFH | | 0EH | --- (unimplemented) |
| 07H (Bank 7) | 1C0-1FFH | | 0FH | --- (unimplemented) |

NOTE: If a value other than those indicated above is used in the "bank" register, an undefined state will result.

User RAM is assigned both in directly addressed register RAM space and in banked register RAM space. Addresses 03AH-07FH (70 bytes) of directly addressed register RAM and Banks 0, A and B (192 bytes) of banked register RAM are assigned for a total of 262 bytes of User RAM.

See the "Special Functions Registers Summary" for details on the contents of SFRs.

## L1 Vector Accelerator/Multiplier

A variety of macros are provided by Sensory that manipulate the L1 Vector Accelerator to provide signed and unsigned multiplication functions. See the "FluentChip™ Technology Library Manual" for information on these macros and their application.

The L1/Multiplier unit may be independently powered down by programming the register D6.Bit 4 to "0" ("clkExt" register, "L1clk_on" bit).

## Digital Filter

The RSC-464 has a Digital Filter engine capable of dividing up a frequency range into several smaller ranges. It is also capable of reporting characteristics of each range to the RSC-464 processor. The configuration of the Digital Filter engine and access to signal characteristics generated are enabled by technology modules that are available from Sensory "Technology Support" upon request.

## Power and Wakeup Control

**NOTE: If developers intend to use sleep or idle mode, they should always use the "GoSleep", "SleepIO", "GoIdle" or "IdleT2" functions/macros provided in the Sensory FluentChip library to ensure proper clock configuration when coming out of sleep or idle mode. Failure to do so may result in some initial instructions being improperly executed after wakeup.**

The typical Active Supply Current is realized when operating with a main clock rate of 14.32 MHz at 3V and all I/O configured to the high-Z state. Lowering clock frequency reduces active power consumption, although FluentChip™ technology typically requires a 14.32 MHz clock.

Two supply current power-down modes are available – Sleep and Idle modes. In Sleep mode everything is stopped, and only an I/O event can initiate a wake-up. In Idle mode OSC2 and Timer2 continue to run, and an Audio Wakeup, I/O Wakeup or Timer2 interrupt request caused by overflow can generate a wake-up.

A low power mode is also available by using Idle with Audio Wakeup. In this mode the Audio Wakeup circuitry will wake the chip from Idle mode. A somewhat higher supply current is consumed in this mode due to the Audio Wakeup logic and an active microphone. An I/O event, Timer2 interrupt request caused by overflow, or audio event can generate a wake-up from this mode. The current consumed by the Audio Wakeup logic is typically 40uA. Combined with OSC2, Timer2 and an active microphone, the total current consumed in Idle with Audio Wakeup mode is about 150 microamps.

**NOTE: If developers intend to use sleep or idle mode, they should always use the "GoSleep", "SleepIO", "GoIdle" or "IdleT2" functions/macros provided in the Sensory FluentChip library to ensure proper clock configuration when coming out of sleep or idle mode. Failure to do so may result in some initial instructions being improperly executed after wakeup.**

Sleep mode is entered by setting register E8.Bit7=1 ("ckCtl" register; "pdn" bit), register E8.Bit0=1 ("osc1_off") and register E8.Bit1=0 (OSC2 off). Idle mode is entered by setting register E8.Bit7=1, register E8.Bit1=1 ("osc2_on") and register E8.Bit0=1. Setting register E8.Bit7=1 ("pdn") freezes the processor, but does *not* insure that the DAC, Audio Wakeup, and the PWM are placed in the lowest possible current-consumption state. Software control must power these blocks down prior to setting "pdn" to "1", according to the procedures indicated in "DAC", "Audio Wakeup", and "Pulse Width Modulator Analog Output" Sections. The "FluentChip™ Technology Library Manual" provides sample code for achieving the lowest current-consumption state for Sleep and Idle modes. The state of "pdn" bit may be observed externally on the PDN pin (see pin definitions in "Package Options" section) and used to control power down of circuitry external to the RSC-464, if desired.

NOTE: GPIO (Ports 0 & 2) should be put in input mode and a known state (e.g. light pull-up) whenever whenever practical to conserve power and not in conflict with their intended function, and especially in powerdown mode to achieve the specified minimum supply current consumption. The external memory interface (A[19:0], D[7:0], -RDR, -WRC, -RDF and –WRD) automatically goes into a high-Z state and is pulled up by a 100 Kohm internal resistor when the "pdn" bit is set, to conserve current.

Register E8 contains both the "pdn" bit and the processor clock selector (Bit2). The clock selector bit determines whether the 14.32 clock ("fast clock") or the 32KHz clock ("slow clock") will be used at wakeup time, independent of what clock rate was being used before or during power down mode. This allows the processor clock *after wakeup* to be the same or different from the processor clock used when the power-down flag was set. (see "Clock" section for complete explanation)

To minimize power consumption, most operational blocks on the chip also have individual power controls that may be selectively enabled or disabled by the programmer.

***Wakeup from powerdown***

**NOTE: If developers intend to use sleep or idle mode, they should always use the "GoSleep", "SleepIO", "GoIdle" or "IdleT2" functions/macros provided in the Sensory FluentChip library to ensure proper clock configuration when coming out of sleep or idle mode. Failure to do so may result in some initial instructions being improperly executed after wakeup.**

During a Wakeup event the processor, which was "frozen" when register E8.Bit7 was set, will be restarted without loss of context. Note that a Wakeup event does not cause a reset. A reset of the chip will also cancel a power down mode, but with a corresponding loss of processor context.

Wakeup events terminate a power-down state. In Sleep mode, only an I/O Wakeup event can initiate a wake-up. In Idle mode, an Audio Wakeup, I/O Wakeup or Timer2 interrupt request caused by overflow can generate a wakeup.

An I/O Wakeup is an edge-triggered event, enabled by setting the bit(s) high in registers E9 corresponding to the desired I/O pin(s) to be used for wakeup. E9 controls P0 wakeup enable. The polarity of the edge causing the Wakeup event is controlled by putting the appropriate port pin in input mode and writing the appropriate bit in the output register for that pin to the desired polarity – a "1" for a positive going edge and a "0" for a negative going edge. (see "General Purpose I/O" section for complete explanation) When the edge on the Wakeup pin matches the polarity assigned in the output register, a Wakeup will occur. When an I/O Wakeup occurs register

FB.Bit1 will be set high. The user should clear this bit once the status is noted, so that it can indicate future wakeup events.

A T2 Wakeup is enabled by setting register E8.Bit6 high. Then an overflow of timer T2 will generate an interrupt request, which in turn will trigger a wakeup event. Note that the Timer2 "irq" bit (register FE.Bit1) must be cleared prior to powering down to allow the wakeup interrupt request to occur. (the "Timers/Counters" section describes how timer T2 is configured)

An Audio Wakeup is generated by special circuitry that can detect several classes of sounds, even while in power-down mode. When the class of sound selected by the programmer is detected by this circuitry a wakeup event will occur. (see the "Audio Wakeup" section for more information)

To determine the source of wakeup during powerdown, it is necessary to query FE.Bit1 and FB.Bit1. If FE.Bit1 is set, then the wakeup was caused by T2. If FB.Bit1 was set, it was caused by I/O. If a wakeup occurs and neither of these bits is set, then by process of elimination the wakeup was caused by Audio Wakeup.

## General Purpose I/O

The RSC-464 has 16 general-purpose I/O pins (P0.0-P0.7 and P2.0-P2.7). Each pin can be programmed as an input with weak pull-up (~200kΩ equivalent device); input with strong pull-up (~10kΩ equivalent device); input without pull-up, or as an output with sufficient drive to light an LED. (See "DC Characteristics" section for I/O electrical characteristics.) This is accomplished by programming combinations of bits of configuration registers assigned to the I/O pins.

**NOTE: Port 1 on the RSC-4128 has been removed on the RSC-464 to reduce cost. If an application began as an RSC-4128 design, it should be reviewed to ensure Port 1 is not being used.**

Two control registers, A and B, are used to control the nature of inputs and outputs for each port. Registers E6 ("p0CtlA") and E7 ("p0CtlB"), and DE ("p2CtlA") and DF ("p2CtlB"), are the control registers A and B for ports P0 and P2, respectively. Each port pin's I/O configuration may be controlled independently by the state of it's corresponding bits in these registers. Control registers A and B together determine the function of the port pins as follows:

| B bit | A bit | Port Pin Function |
|---|---|---|
| 0 | 0 | Input - Weak Pull-up |
| 0 | 1 | Input - Strong Pull-up |
| 1 | 0 | Input - No pull-up |
| 1 | 1 | Output |

(For example, if register E7.Bit 4 is set high, and register E6.Bit 4 is low, then pin P0.4 is an input without a pull-up device.)

After reset, pins P0.0-P0.7 and P2.5-P2.7 are set to be digital inputs with weak pull-ups, and pins P2.0-P2.4 are configured as analog input pins with no pull-ups. Being reset as an input and lightly pulled to a known (high) state ensures minimum power consumption as a default beginning. Eight of these pins (Port P0) can also be configured as inputs to control IO Wakeup events. (see "Power and Wakeup Control" section).

P2.0, P2.1, P2.3, and P2.4 can be configured as comparator inputs. P2.2 can be configured as a comparator reference. Some or all of P2.0-P2.4 can be configured as digital inputs by the use of the "cmpCtl" register (register D4) Bits[2:0] (see "Comparator Unit" section)
Note: When configuring P2.0-P2.4 as digital inputs the associated weak pull-up should be selected as shown above.

P0.0 and P0.2 can be configured as External Interrupts (see "Interrupts" section). P0.1 can be configured in input mode as a gate for an external event counter. (See "Timers/Counters" Section)

Registers E5 (“p0In”) and E4 (“p0Out”), and DD (“p2In”) and DC (“p2Out”), provide paths for data input and data output on P0 and P2, respectively. The port input registers (E5, E1, and DD) are actually buffers that record the value at the ports at the time they are read. The port output registers (E4, E0 and DC) latch the data written to them and express it on the ports when the ports are configured as an output.

Following is a summary of the general purpose I/O control registers:

| Register | | |
|---|---|---|
| 0DCH | Read/Write | P2[0:7] (port 2) output register. Cleared by reset. |
| 0DDH | Read | Port 2 input. |
| 0DEH | Read/Write | Port 2 Control Register A. Cleared by reset. |
| 0DFH | Read/Write | Port 2 Control Register B. Bits[7:5] cleared by reset.<br>Bits[4:0] set by reset |
| 0E4H | Read/Write | P0[0:7] (port 0) output register. Cleared by reset. |
| 0E5H | Read | Port 0 input. |
| 0E6H | Read/Write | Port 0 Control Register A. Cleared by reset. |
| 0E7H | Read/Write | Port 0 Control Register B. Cleared by reset. |

***GPIO during powerdown***

GPIO should be put in input mode and a known state (e.g. light pull-up) whenever practical to conserve power and not in conflict with their intended function, and especially in powerdown mode to achieve the specified minimum supply current consumption.

## Memory Addressing

The RSC-464 can address up to 64Kbytes of default internal ROM providing Constant/Code Space. Constant/Code Space is read-only in the RSC-464

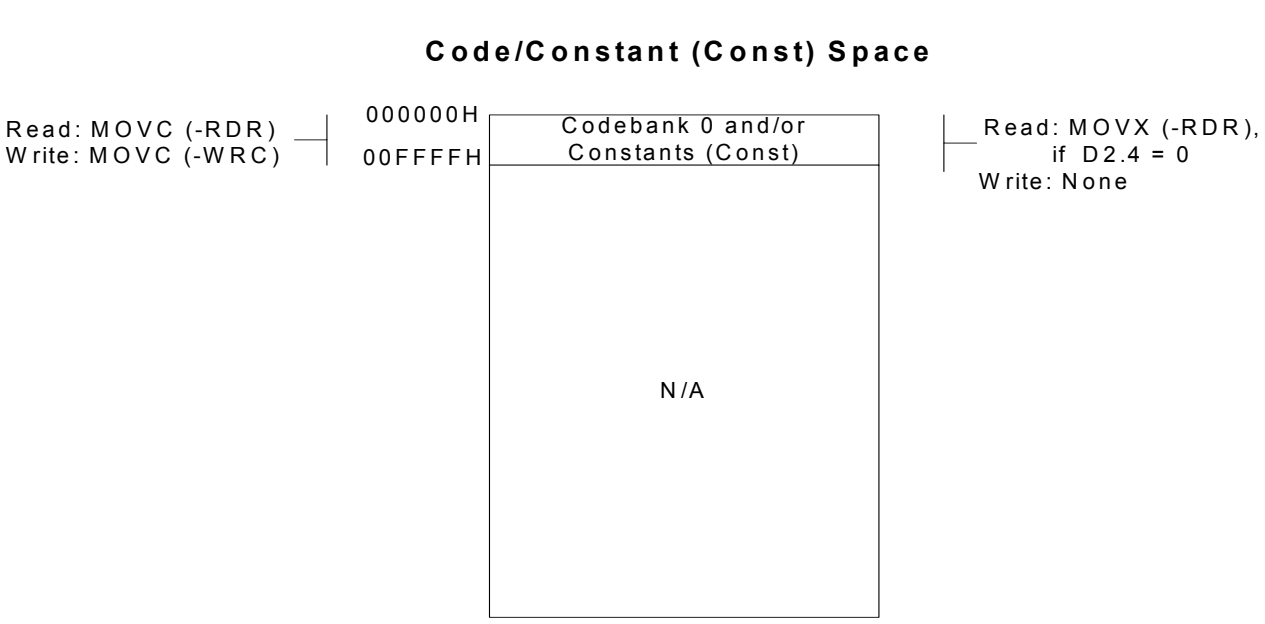
One may also interface to serial memory devices for storage and retrieval of speech data, by using the serial drivers for ROM, Flash, EEPROM, etc. provided in the FluentChip™ Technology Library. The serial memory option is useful for applications for which speech or music data exceed the storage capacity of on chip ROM. The specific I/O used by the serial interface are configurable. (See the "FluentChip™ Technology Library Manual" for more information). An example of the optional use of external serial Flash is provided in Reference Schematic 1-1.

***Constant/Code Space***

When reading Constant/Code Space, an application can access up to 64KBytes. The *MOVC* and *MOVX* instruction can read these first 64KBytes. However, the *MOVC* is more efficient for reading Constants within the current Code Bank. This 64KBytes is called Code Bank 0.

NOTE: Constant Space may be referred to as "Const Space" in assemblers and compilers.

## Memory Map Diagram




## Oscillators

Two independent oscillators in the RSC-464 provide a high-frequency oscillator (OSC1), and a 32 KHz time-keeping and power-saving oscillator (OSC2). The oscillator characteristics are:

| OSC | FREQ | PLL | PINS | SOURCES |
|---|---|---|---|---|
| **1** | 3.58 MHz | 4X | XI1<br>XO1 | Crystal<br>Ceramic resonator<br>LC |
| **2** | 32768 Hz | N/A | XI2<br>XO2 | Crystal<br>Internal RC |

***OSC1***

OSC1 is enabled by programming register E8.Bit0 to "0", which is the reset state for this bit. This bit is also programmed to "0" during a Wakeup Event, enabling OSC1, if register E8.Bit2 is programmed to "0". (see "Power and Wakeup Control" section) In this case, a 10-20 millisecond delay will be forced to allow OSC1 to reach stable oscillation. OSC1 must run at 3.58 MHz when using the FluentChip™ technologies, but may be slower if the RSC-464 is used as a general purpose platform for other applications. When OSC1 is disabled, the PLL which generates the 14.32MHz clock (CLK1) is also disabled.

***OSC2***

OSC2 is enabled by programming register E8.Bit1 to "1". The reset state for this bit is "0", so this oscillator is disabled by reset. OSC2 will be enabled during a Wakeup Event if register E8.Bit2 is programmed to "1". (see "Power and Wakeup Control" section) No delay will be forced, as OSC2 is assumed to be running during Idle mode. The OSC2 source may be set to an external 32 KHz crystal by programming register EF.Bit2 to "0" (Note: register EF.Bit7 must be "0" to enable writing EF.Bit2). The external 32 KHz crystal should be used when accurate timing and/or time-keeping is essential. In this mode, OSC2 is capable of achieving errors as low as 20ppm, depending on the quality of the crystal and crystal circuit design. A typical value for the crystal bias capacitors is 27pF, but this will vary depending on the crystal quality and stray capacitance inherent in the application board layout.

The OSC2 source may be set to an on-chip RC by programming register EF.Bit2 to "1" (Note: register EF.Bit7 must be "0" to enable writing EF.Bit2). When using the on-chip RC, no external components are required for OSC1. The on-chip RC value will vary due to process, temperature and supply voltage variations, so this oscillator frequency will vary by +/- 30%. The on-chip RC mode should be used for low power modes where timing is not critical and minimum system cost is important.

***Oscillator Stabilization***

When exiting Sleep mode (see "Power and Wakeup Control" section) OSC1 will have a forced 10-20millisecond delay for stabilization if it is enabled. If OSC2 is enabled, it may require several seconds to stabilize, after which the RSC-464 will begin running. Therefore, for fast response out of Sleep mode OSC1 should be enabled.

## Clocks

The RSC-464 uses a fully static core – the processor can be stopped (by removing the clock source) and restarted without causing a reset or losing contents of internal registers. Dynamic operation is guaranteed from ~1KHz to 14.32 MHz.

***Fast Clock***

The 3.58 MHz OSC1 frequency is quadrupled by an on-chip PLL to produce a 14.32 MHz internal clock (CLK1). Creating the internal clock in this way avoids an expensive high frequency crystal, substantially reducing overall system cost. When used as the processor clock (see below), the 14.32 MHz internal clock creates internal RAM cycles of 70 nsec duration, and internal or external Code/Data memory cycles of 140 nsec duration**.** Careful design may allow operation with memories having access times as slow as 140 nsec.

***Slow Clock***

OSC2 generates an internal clock (CLK2) with an equivalent frequency to OSC2. When used as the processor clock (see below), the RAM access cycles are one CLK2 cycle and Code/Data access cycles are two CLK2 cycles.

***Processor Clock***

Either CLK1 or CLK2 can be selected as the processor clock (PCLK) on the fly by changing the value of register E8.Bit2. The reset state defaults to CLK1. (NOTE: it is possible to select a disabled clock as the processor clock. It is the responsibility of the programmer not to select a clock until the corresponding oscillator has been enabled and allowed to stabilize.) Power savings may result by using CLK2 when the processor is a lower activity mode and using CLK1 when in a higher activity mode. If the use of an external clock driver is desired, the output of that driver should be connected to the XI1 pin.

After source selection, the processor clock can be divided-down in order to limit power consumption. Register E8.Bits 4 and 3 determine the divisor:

| E8.Bit4 | E8.Bit3 | Processor Clock Divisor |
|---|---|---|
| 0 | 0 | 1/2 |
| 0 | 1 | 1/1 (reset default) |
| 1 | 0 | 1/8 |
| 1 | 1 | 1/256 |

A Processor Clock divisor of 1/1 is typically required for FluentChip™ technology.

The processor clock is gated by the Wake-up delay and also gated by "pdn"=0 (register E8.Bit7), in such a way that the processor is stopped in a zero-power state with no loss of context.

***Other System Clocks***

The following functional clocks are generated from OSC1: CLK1, the digital filter clock, the analog front end (AFE) master clock, the L1 clock, Timer1 clock, Timer3 clock, and the Multi-Task timer clock. The Timer2 clock and the Watchdog timer clock are generated from OSC2. (see each block's section for clocking details) All clocks except the Timer2 and Audio Wakeup clocks are gated with the pdn = 0, to assure they are disabled during IDLE and SLEEP modes. Timer2 and Audio Wakeup can run during Idle mode to produce a T2 Wakeup or Audio Wakeup. (see "Power and Wakeup Control" section)

## Timers/Counters

Four programmable timers and one fixed timer in the RSC-464 provide a variety of timing/counting options. Timers 1, 2, 3 and the Multi-Tasking timer can all generate interrupts upon overflow. (See "Interrupts" section)

***Timers 1 and 3***

Each of Timer1 (T1) and Timer3 (T3) consists of an 8-bit reload value register, an 8-bit up-counter, and a 4-bit decoded prescaler register. Each is clocked by CLK1 divided by 16. The reload register is readable and writeable by the processor. The counter is readable with precaution taken against a counter change in the middle of a read.

NOTE: If the processor writes to the counter, the data is ignored. Instead, the act of writing to the counter causes the counter to preset to the reload register value.

When the timer overflows from FFH, a pulse is generated that sets register FE.Bit 0 ("irq" register; T1 bit) or register FE. Bit 4(T3 bit). The width of the pulse is the pre-scaled counter clock period. Instead of overflowing to 00, the counter is automatically reloaded on each overflow.

For example, if the reload value is 0FAH, the counter will count as follows:

0FAH, 0FBH, 0FCH, 0FDH, 0FEH, 0FFH, 0FAH, 0FBH etc.

The overflow pulse is generated during the period *after* the counter value reaches 0FFH.

A separate 4-bit decoded prescaler register is between the clock source and the up-counter for each of T1 and T3. The 4bits represent the power of 2 used to divide the timer clock before applying it to the up-counter. For example, a prescaler value of 0 passes the timer clock directly through (divides by 2^0 = 1). A prescaler value of 5 divides the timer clock by 2^5 = 32.

| Prescaler value | Divisor | Prescaler value | Divisor |
|---|---|---|---|
| 0000 | 1 | 1000 | 256 |
| 0001 | 2 | 1001 | 512 |
| 0010 | 4 | 1010 | 1024 |
| 0011 | 8 | 1011 | 2048 |
| 0100 | 16 | 1100 | 4096 |
| 0101 | 32 | 1101 | 8192 |
| 0110 | 64 | 1110 | 16384 |
| 0111 | 128 | 1111 | 32768 |

The resolution of T1 and T3 is 8 bits, but the range is 23 bits. The longest interval that can be timed by T1 or T3 is 2^15*256 clocks = 9.3 seconds.

The 4-bit prescaler for T1 is in the Clock Extensions Register, (register D6.Bits[3:0]). The 4-bit prescaler for T3 is in the Timer3 Control Register (register D9.Bits[3:0]).

In addition to its timing capability, T3 can also be configured as a counter of external events. In this configuration it uses either the rising or falling edge of a signal applied to I/O pin P0.1. The selected transition is internally synchronized to CLK1. The maximum external count rate for T3 is 447KHz.

The Timer3 Control Register contains the counting/timing options for T3. The register is write-only. Bits[6:4] provide configuration control.

| Bit6 | Bit5 | Bit4 | timer source | Configuration |
|---|---|---|---|---|
| x | 0 | 0 | T3CLK | timer |
| 0 | 0 | 1 | T3CLK | timer gated by P0.1 LOW |
| 1 | 0 | 1 | T3CLK | timer gated by P0.1 HIGH |
| 0 | 1 | x | P0.1 | count P0.1 events, rising edge |
| 1 | 1 | x | P0.1 | count P0.1 events, falling edge |

Bit 7 0: disable T3 and prescaler from counting/timing
1: enable T3
cleared by reset.

Bit 6 0: use rising edge for external event counting
use LOW state on pin P0.1 for timer gating
1: use falling edge for external event counting
use HIGH state on pin P0.1 for timer gating
cleared by reset

Bit 5 0: use internal T3CLK for source (timing)
1: use external events on pin P0.1 for source (counting)
cleared by reset

Bit 4 0: normal operation
1: T3 is gated by pin P0.1 according to Bit6
cleared by reset.

Bit 3:0 Encoded prescaler for T3. (See prescaler table above).
cleared by reset.

T1 and T3 can generate interrupts upon overflow by setting register FD.Bit0=1 and Bit4=1, respectively. (see "Interrupts" section)

***Timer2***

Timer2 (T2) is clocked by CLK2 divided by 128. The overflow pulse from T2 can cause an interrupt request which in turn will cause a T2 Wake-up from power-down, if register E8.Bit6=1. (see "Power and Wakeup Control" section). Note that the Timer2 "irq" bit (register FE.Bit1) must be cleared prior to powering down to allow the wakeup interrupt request to occur. T2 can also generate a standard interrupt request by setting register FD.Bit1=1. (see "Interrupts" section)

***Timers 1, 2 and 3 Timer Reload and Counter Registers***

All are cleared to zero on reset.

| Register | addr | | |
|---|---|---|---|
| t1r | EBH | Read/Write | Timer1 Counter Reload (2's complement of period) |
| t1v | ECH | Read | Timer1 current counter value |
| | | Write | Force load of Timer1 counter from reload register |
| t2r | EDH | Read/Write | Timer2 Counter Reload (2's complement of period) |
| t2v | EEH | Read | Timer2 current counter value |
| | | Write | Force load of Timer2 counter from reload register |
| t3r | DAH | Read/Write | Timer3 Counter Reload (2's complement of period) |
| t3v | DBH | Read | Timer3 current counter value |
| | | Write | Force load of Timer3 counter from reload register |

***Multi-Task Timer***

The multi-tasking (MT) timer is intended to count a fixed interval of 858.1 microseconds. This provides a "heartbeat" for multi-tasking in the FluentChip™ technology library. Other applications may find this useful for similar purposes. This interval is obtained by dividing the CLK1 rate, when running at 14.32 MHz, by a fixed factor of 12288. There is no configurability to the MT timer. One bit in the Clock Extension Register (D6.Bit6) enable this timer's clock. The MT timer overflow can generate an interrupt by setting register FD.Bit7=1. (see "Interrupts" section)

***Watchdog Timer***

Due to static electricity, voltage glitches, or other environmental conditions (or program bugs!), a software program can begin to operate incorrectly. The watchdog timer provides protection from such errant operation.

The Watchdog Timer (WDT) unit comprises two control bits in the System Control Register (D5), a special instruction, two status bits, and a 17-bit counter. The counter, driven by OSC2, produces a toggle rate of approximately 4 seconds at the 17$^{th}$ bit. A 2-bit decoded mux in the "sysCtl" register (register D5) allows selecting the WDT timeout pulse from bit 9, 13, 15, or 17 of the counter. This selection sets the timeout in the range of approximately 15.6 msec to 4 seconds. The accuracy of these times will depend on whether the OSC2 source is a 32 KHz crystal or the on-chip RC.

The WDT is enabled by register FB.Bit4=1. This bit can only be set by execution of the "WDC" instruction. This bit is cleared by reset, so the WDT is disabled by reset. The bit is also cleared when E8.Bit7=1 (pdn), so the WDT is disabled in either SLEEP or IDLE mode. It is not automatically re-enabled on Wakeup. Program control cannot write to register FB.Bit4 to enable or disable the WDT. That is, FB.Bit4 is a read-only bit for normal register access instructions. Since the WDT needs OSC2 for its operation, once the WDC instruction has been executed and register FB.Bit4=1 to enable the WDT, OSC2 cannot be disabled by programming register E8.Bit1 =0 unless the "pdn" bit (register E8.Bit7) is also *set* simultaneously. This allows disabling the WDT only when entering a power down mode and is intended to reduce the probability of accidental software disabling of the WDT in active mode.

Executing the WDC instruction clears the WDT counter, sets register FB.Bit4=1, clears register FB.Bit5=0 (wd_timed-out), and starts a new timeout period. The OSC2 oscillator may also be enabled by executing the WDC instruction. If the oscillator is stopped, executing this instruction also sets register E8.Bit1=1 to enable OSC2. In this case, timing will not begin until the oscillator is active.

Once the WDT is started, software must execute the WDC instruction at a rate faster than the timeout period. Otherwise the watchdog circuit sets the "watch dog timed out" bit (register FB.Bit5) and generates a Timed Out Reset, which resets the RSC-464. A Timed Out Reset disables the WDT. (See "Reset" section) Software in the reset routine can detect that the WDT timed out (FB.Bit5=1), since that is preserved during the Timed Out Reset. Placing the chip in Sleep or Idle mode disables the WDT operation.

***Timer Powerdown***

Some timers have independent power down control, while others may only be powered down by turning off their clock source, setting the "pdn" bit, or resetting. It is not required for the application to do this for full chip power down, as long as it complies with directions in the "Power and Wakeup Control" section. However, one may choose to reduce power consumption in active mode by turning off individual timers.

Timer 3 and MT Timer may be independently powered down by setting the register D9.Bit 7 to "0" ("t3Ctl" register, "t3_on" bit) and register D6.Bit 6 to "0" ("clkExt" register, "MTclk_on" bit), respectively.
Timer 1, Timer 2 and the WDT require special circumstances to powerdown, which are appropriate for their application. See their respective descriptions for more detail.

## Interrupts

The RSC-464 allows for 8 interrupt request sources, as selected by software. All are asynchronous positive edge activated except the two external requests, which have programmable edges. Each has its own mask bit and request bit in the "imr" and "irq" registers respectively. There is a Global Interrupt Enable flag in the "flags" registers. The "imr" and "irq" bits are listed below with the RSC-464 interrupt source shown in parenthesis:

0FDH "imr"

Bit 7: 1= enable interrupt request #7 (Overflow of MT timer)
Bit 6: 1= enable interrupt request #6 (Edge of P0.2)
Bit 5: 1= enable interrupt request #5 (Block End)(Reserved for Technology code)
Bit 4: 1= enable interrupt request #4 (Overflow of Timer3)
Bit 3: 1= enable interrupt request #3 (Edge of P0.0)
Bit 2: 1= enable interrupt request #2 (Filter End Marker)(Reserved for Technology code)
Bit 1: 1= enable interrupt request #1 (Overflow of Timer2)
Bit 0: 1= enable interrupt request #0 (Overflow of Timer1)

0FEH "irq"

Bit 7: 1=interrupt request #7 (Overflow of MT Timer)
Bit 6: 1= interrupt request #6 (Edge of P0.2)
Bit 5: 1=interrupt request #5 (Block End)(Reserved for Technology code)
Bit 4: 1= interrupt request #4 (Overflow of Timer3)
Bit 3: 1= interrupt request #3 (Edge of P0.0)
Bit 2: 1= interrupt request #2 (Filter End Marker)(Reserved for Technology code)
Bit 1: 1= interrupt request #1 (Overflow of Timer2)
Bit 0: 1= interrupt request #0 (Overflow of Timer1)

If an “irq” bit is set high and the corresponding “imr” bit is set high and the Global Interrupt Enable (“gie”; register FF.bit0) bit is set high, an interrupt will occur. Interrupts may be nested if software handles saving and restoring the “flagsHold” register (register CF). The “flags” register is copied to the “flagsHold” register and then the Global Interrupt Enable is cleared, preventing subsequent interrupts until the IRET instruction is executed. The IRET instruction will restore the “flags” register from the “flagsHold” register. The Global Interrupt Enable bit in the “flags” register must not be re-enabled during the period after an interrupt has been acknowledged and before an IRET instruction has been executed unless interrupt nesting is desired.

If an interrupt occurs during an instruction that clears the Global Interrupt Enable bit (typically the CLI instruction) the value of the “gie” bit will be 0 upon completion of the Interrupt Service Routine and Return From Interrupt to the instruction following the one that cleared the “gie” bit. (NOTE: This is a change from the operation of the RSC-364.)

ERRATA NOTE: “Shadowing” of the “gie” flag bit in the flagsHold register is not disabled during an Interrupt Service Routine (ISR) as intended, due to design errata. Therefore, any operation that directly modifies the flags register “gie” bit (mov, logic operations, arithmetic operations, CLI, STI, etc directly on the flags register) will erroneously invoke a “shadowing” of the flags register “gie” bit in the flagsHold register.
To avoid this problem, simply store the flagsHold register in a temporary location immediately upon entering an ISR and restore the flagsHold register from that temporary location as the final instruction before returning from the ISR.

The “flagsHold” register is accessible under program control at address CF in order to improve multi-tasking operation.

External interrupts may be enabled on pins P0.0 (1st external interrupt request) and P0.2 (2nd external interrupt request), by setting register FD.Bit3=1 and register FD.Bit6=1, respectively. The polarity of the edges to trigger an external interrupt request for P0.0 and are controlled by register D5.Bits[1:0]. Setting D5.Bit0=0 will cause a positive going edge on P0.0 to generate and interrupt and D5.Bit0=1 will cause a negative going edge to generate an interrupt. The same controls for P0.2 are possible with D5.Bit1. The corresponding external “irq” flag will be set if the transition matches the interrupt edge control bit.

NOTE: If P0.0 or P0.2 are configured as outputs, writing to those outputs can trigger external interrupt requests if the proper edge polarities occur. The user must be careful to avoid this, unless it is intended to use this as a way of generating interrupt requests under internal software control.

An interrupt is disabled by writing a zero to the corresponding bit in the imr register (register 0FDH). However, an active interrupt request can still be pending. To be certain that an interrupt does not happen, you should clear the interrupt request flag in the irq register (register 0FEH) as well. For example:

```
; Disable timer 1 interrupt

cli
and     imr,#0FEH       ; mask new interrupt requests
mov     irq,#0FEH       ; clear any pending interrupt request
sti
```

For each interrupt, execution begins at a different address:

| | | |
|---|---|---|
| Interrupt #0 | Address 04H | (Overflow of Timer 1) |
| Interrupt #1 | Address 08H | (Overflow of Timer 2) |
| Interrupt #2 | Address 0CH | (Filter End Marker)(Reserved for Technology code) |
| Interrupt #3 | Address 10H | (Edge of P00) |
| Interrupt #4 | Address 14H | (Overflow of Timer 3) |
| Interrupt #5 | Address 18H | (Block End)(Reserved for Technology code) |
| Interrupt#6 | Address 1CH | (Edge of P02) |
| Interrupt#7 | Address 20H | (Overflow of MT timer) |

The interrupt vector is generated as a 20-bit address. The low 16 bits are derived from the execution table above, and the high 4 bits are selected as a normal code fetch as described in the “Memory Addressing” section. Specifically, the “cb1” bit is not touched by the interrupt.

If the corresponding mask register bit is clear, the “irq” bit will not cause an interrupt. However, it can be polled by reading the “irq” register.

"irq" bits can be cleared by writing a "0" to the corresponding bit at register FE (the "irq" register). "irq" bits cannot be set by writing to register FE. Writing a "1" to that register is a NO-OP.

The "irq" bits must be cleared within the interrupt handler by an explicit write to the "irq" register rather than by an implicit interrupt acknowledge.

**PLEASE NOTE:**

Clear interrupts this way –

```
mov irq, #bitmask        ; CORRECT
```

Not this way –

```
And irq, #bitmask        ; INCORRECT
```

The "and" instruction is not a single action. The "and" instruction is a read-modify-write action. If an interrupt occurs during an "and irq" operation the interrupt will be cleared before it is seen, possibly disabling the interrupt until the system is reset. Because one cannot directly set or clear bits in the "irq" register, use "mov irq" as a safe and effective single action to clear bits in the "irq" register. Use it the way you would use an "and" instruction to operate on other registers.

NOTE: Bit2 and Bit5 of the "irq" register should always be written as "1" when clearing other "irq" bits, to avoid conflicts with the Technology code use of these bits.

In Idle mode, Timer2 continues to operate even when the rest of the RSC-464 is powered-down. An overflow from Timer2 will set the corresponding "irq" flag even when there is no clock input to the processor. Note that the Timer2 "irq" bit (register FE.Bit1) must be cleared prior to powering down to allow the wakeup interrupt request to occur. This may also lead to normal interrupt processing once the processor is active, if the Timer 2 "imr" bit is set (register FD.Bit1). This interrupt response is unique from, and may be in addition to, the T2 Wakeup.

## Analog Input

The analog front end (AFE) for the RSC-464 consists of a preamplifier with gain control, a 16-bit analog-to-digital converter, digital decimator and channel filters, and associated references. A single analog input can be processed through the AFE. All of this circuitry can be powered down to conserve battery life by programming register EF.Bit0 to "0". Setting this bit to "1" powers up the circuitry, requiring a settling time of approximately 10milliseconds.

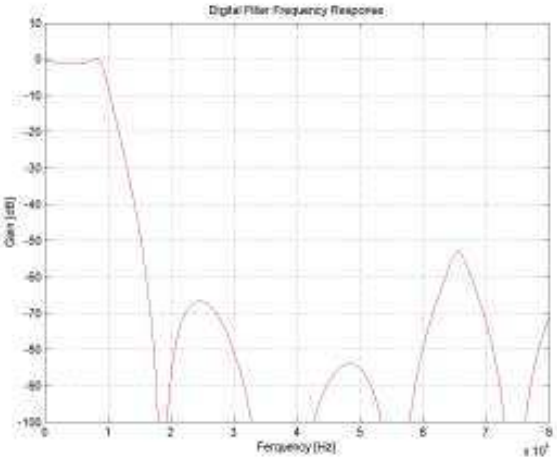
The analog front end (AFE) performs analog to digital conversions on a low-level signals, which may be derived from an electret microphone. The microphone signal is amplified by a preamp that provides four levels of gain, which are selected by programming register D5.Bits[4:3]. Full-scale output for the four settings corresponds to input signals of 100, 50, 25, and 12.5 millivolts Vpeak-peak, as shown in the table below.

| Gain | Input Referred Noise | Max Input Signal | |
|---|---|---|---|
| "sysCtl" Bits[4:3] | μVrms | mVp-p | mVrms |
| 00 | 5.2* | 100 | 35.4 |
| 01 | 4.9* | 50 | 17.7 |
| 10 | 4.6* | 25 | 8.8 |
| 11 | 4.4 | 12.5 | 4.4 |

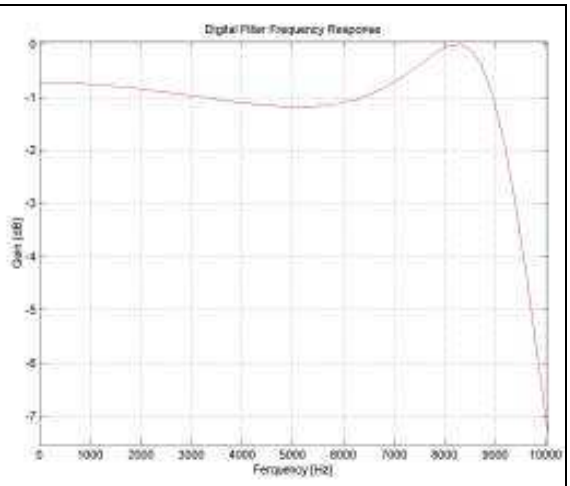
Input signals higher than specified will produce a saturated full scale output with no wrap around. A line level audio input must be attenuated to the range shown above for use with the AFE.

**Digital Transfer Functions**




Lowpass response




Detail of passband

| Frequency | Attenuation | |
|---|---|---|
| | Min | Max |
| Below 8 kHz | 0 | 1.18 |
| 9.395 kHz | 3 dB | |
| 20 kHz | 87.82 | |
| Above 20 kHz | 53 | |

NOTE: A 1uF capacitor should be connected to AMPCOM and tied to GND, a 2.2uF should be connected to VCM and tied to GND, and a 0.1uF capacitor should be connected to VREF and tied to GND. Failure to connect this capacitors will substantially degrade ADC performance, and FluentChip™ technology.

***A/D Conversion***

The amplified signal is processed by a delta-sigma A/D converter that provides a 1-bit over-sampled digital signal. This digital stream is filtered and decimated to produce 16-bit samples at the fixed rate of 18,636 samples per second. The 16 bit signal will have about 12.5 bits of dynamic range, with about 10 bits above the noise level. These samples are then provided to the RSC-464 digital filter unit formatted as signed two's-complement 16-bit values. The samples are stored in the digital filter input registers "adcSampleHi" (register F5) and "adcSampleLo" (register F4).

Note: Using the AFE for general purposes other than as intended in FluentChip™ technology modules may conflict with FluentChip™. Such conflicts may adversely impact FluentChip™ functionality and/or the functionality of the general purpose application. Care should be taken to avoid such conflicts. Contact Sensory Technical Support for help in this area.

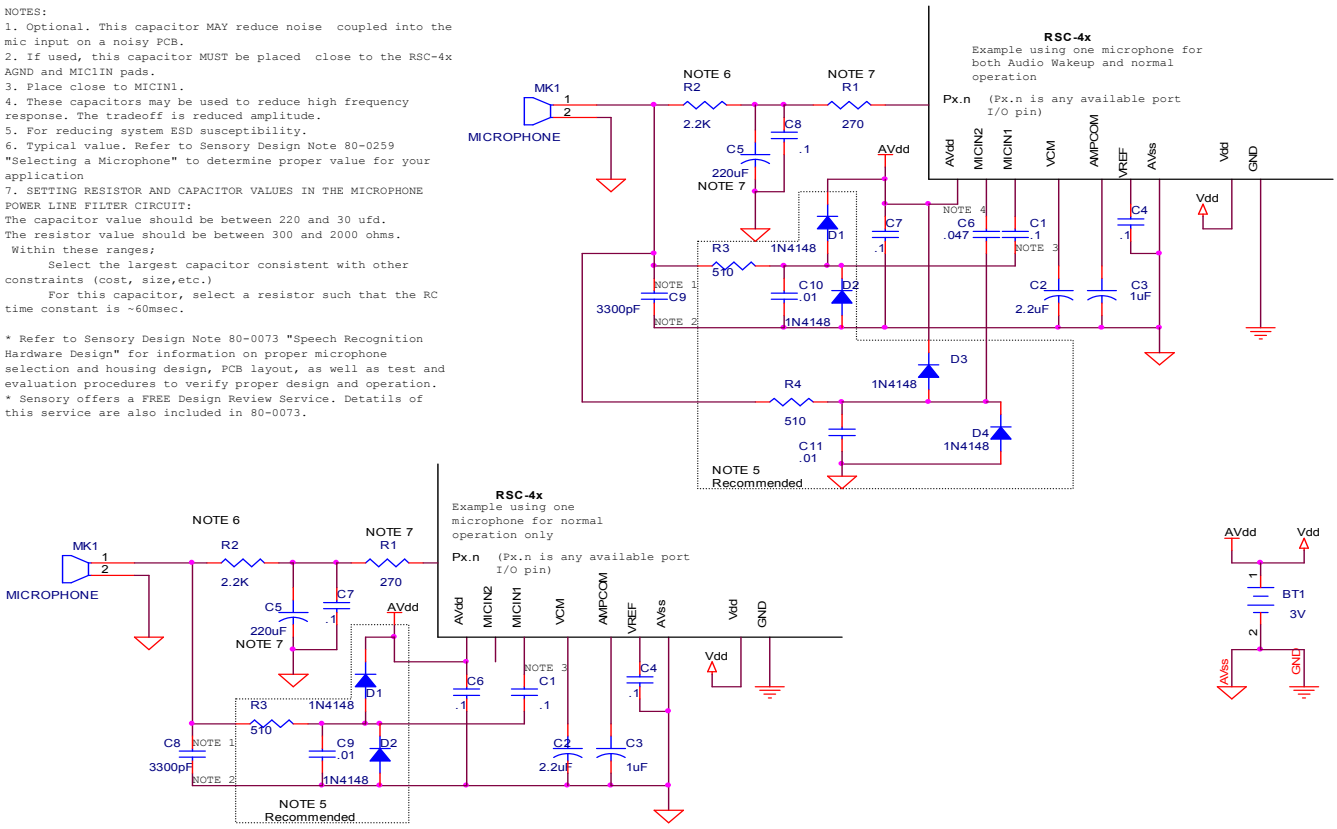
## Audio Wakeup

The Audio Wakeup unit is an analog/digital circuit that can be configured to wakeup from one of four specific audio events:

1) Two handclaps, or any two sharp, closely spaced sounds
2) Three handclaps, or any three sharp, closely spaced sounds
3) A whistle
4) Any "loud" sound above a specified amplitude, with duration options of 1 or 2 seconds

Because it is intended to “listen” continuously at very low power levels, the Audio Wakeup unit must detect each of these events without any processor interaction. The processor configures and enables the unit under program control before going into Idle mode. Audio Wakeup is not available in Sleep mode because the unit requires the CLK2 signal. The detection signal from the Audio Wakeup unit can trigger a wakeup event, which starts the processor and allows further audio processing. The processor inputs to the Audio Wakeup are an enable signal and control signals to select for which sound to listen. See schematic 1-2 for details on this implementation.

**Schematic 1-2**




The RSC-464 FluentChip™ library contains routines for detecting each of the four audio events listed above. These routines also manage powerdown appropriately. See the “FluentChip™ Technology Library Manual” for reference code to invoke these routines.

## Microphones

A single electret microphone may be used both for the analog front-end input (for recognition purposes) and as the sound source for the Audio Wakeup unit. The current consumption and frequency response requirements are different for the two uses, so two microphone input pads are provided: MICIN1 for the normal recognition input to the analog front-end, and MICIN2 for the Audio Wakeup analog front end. A common microphone ground is used for both the normal recognition analog front-end and the Audio Wakeup analog front end.

During normal recognition and Audio Wakeup operation, the microphone would typically be powered from a source with an impedance in the range of 1-2 Kohms. If both the normal recognition and Audio Wakeup front ends are used, they must be isolated from each other by capacitors and may share one microphone and microphone bias circuit. The switching of the microphone input source is under program control. See schematic 1-2 for details on this implementation.

The recommended value for the microphone filter capacitor (labeled “C5” in Schematic 1-2) is in the range of 33uF-220uF. Using a capacitor at the upper end of this range will reduce low frequency noise. Low frequency noise on the microphone input typically won’t affect recognition, but could affect the quality of speech playback when using Record and Playback technology in an application. (see the “FluentChip™ Technology Library Manual” for more information on Record and Playback) Typical low frequency noise sources include 60 Hz hum,

"motor boating" or cyclical fluctuations in the system power supply from "sagging" due to flash writes during speech recording, and LED blinking during recording of speech. All of these effects are reduced in speech playback by using a capacitor closer to 220uF.

**NOTE:** See Design Notes - "Microphone Housing" and "Selecting Microphone" on the RSC-4x Demo/Evaluation CD. Improper microphone circuit and/or enclosure design will result in poor recognition performance.

## Reset

An external reset is generated by applying a low condition for at least two clock cycles on -RESET, an active low Schmitt trigger input. The output of the Schmitt trigger passes through a 10 nsec glitch blocking circuit, followed by an asynchronous flip-flop. The output of the flip-flop generates active high reset throughout RSC-464. The internal reset state is held for 20 msec (when clocked by a 14.32 MHz PCLK). The purpose is to allow the oscillator to stabilize and the PLL to lock before enabling the processor and the other RSC-464 circuits.

During power up, there is an additional delay of up to 10 msec while the oscillator achieves a large enough voltage swing for the logic to begin clocking. This is added to the 20 msec delay described above. So the internal reset state may be held for a maximum total of 30 msec for a reset during power up.

External reset clears the Global Interrupt Enable flag and begins execution at address 0h. The special function registers will be cleared, set, or left as-is, as detailed in the "Special Function Registers Summary" section.

### *Watchdog Timeout Reset*

A special Watchdog Timeout Reset is produced if the Watchdog Timer is enabled and the Watchdog counter times out. The only difference between the Watchdog Timeout Reset and an ordinary reset is that the "wd_timed" bit in the "sysStat" register (register FB.Bit5) is preserved as "1" for a Watchdog Timeout Reset

## Digital-to-Analog-Converter (DAC) Output

The DAC consists of an R-2R network with 10 bits of resolution and an output impedance of approximately 11 Kohms. An external amplifier is required to drive a speaker when using the DAC. The specifications of that amplifier will determine the best choice of speaker impedance and the resulting volume.

The 10-bit resolution corresponds to an analog voltage range between 0V and Vdd minus 1 LSB (represented as "Vdd-"). At Vdd=3V, one LSB of the R-2R network corresponds to about 3 mV. For example:

| R2R Value | DAC output; Vdd=3V |
|---|---|
| 000H = 0v | 0.000V |
| 001H = 0v+ | 0.003V |
| 200H = Vdd/2 | 1.500V |
| 3FFH = Vdd- | 2.997V |

There are two DAC output modes, full-scale and half-scale. In full-scale mode the output voltage swings between 0v and Vdd-; in half-scale mode the output swings between Vdd/4 and 3Vdd/4 minus 1 LSB (roughly Vdd/2 +/- Vdd/4). Values written into the DAC hold register and certain Analog Control register bits are converted into analog voltages.

The DAC hold register ("dac"; register FA) presents an 8-bit *signed* value to the DAC unit. In full-scale mode, the 8 most significant bits are driven by the DAC hold register and the 2 least significant bits are driven by the LSB1 and LSB0 bits in the Analog Control register ("anCtl"; register EF.Bits[5:4]). This results in a total output range of –512 to +511. In half-scale mode the 8 middle bits of are driven by the DAC hold register, the most significant bit is generated automatically by sign extension, and the least significant bit is driven by bit LSB1 in the Analog Control register. This gives a total output range of –256 to +255. The half-scale mode is enabled by setting the mode bit (d2a_half) equal to "1" in register EF.Bit3. The tables below show a selection of values and the resulting output voltage.

**Note: Register EF.Bit7 ("-anctlen") must be "0" in the value being written to register EF, when writing EF.Bit2.**

**Full-Scale Mode** (Output range 0v to Vdd- 1 LSB)

| Decimal Equivalent | DAC hold reg[7:0] (hex) | Analog Cntrl [5:4] (binary) | Digital input | Analog Voltage output | |
|---|---|---|---|---|---|
| | | | | General | 0-3V (approx) |
| -512 | 80H | 00 | 000H | 0V | 0.000V |
| -511 | 80H | 01 | 001H | 0V+ 1 LSB | 0.003V |
| -510 | 80H | 10 | 002H | | 0.006V |
| -509 | 80H | 11 | 003H | | 0.009V |
| -508 | 81H | 00 | 004H | | 0.012V |
| | | | | | |
| -2 | FFH | 10 | 1FEH | | |
| -1 | FFH | 11 | 1FFH | Vdd/2- 1 LSB | 1.497V |
| 0 | 00H | 00 | 200H | Vdd/2 | 1.500V |
| +1 | 00H | 01 | 201H | Vdd/2+ 1LSB | 1.503V |
| +2 | 00H | 10 | 202H | | |
| +3 | 00H | 11 | 203H | | |
| +4 | 01H | 00 | 204H | | |
| | | | | | |
| +510 | 7FH | 10 | 3FEH | | 2.994V |
| +511 | 7FH | 11 | 3FFH | Vdd- 1LSB | 2.997V |

The translation in Full-Scale mode is:
R2R[9] = dac[7] inverted
R2R[8:2] = dac[6:0]
R2R[1:0] = anCtl[5:4]

**Half-Scale Mode** (Output range Vdd/4 to 3Vdd/4- 1 LSB)

| Decimal Equivalent | DAC hold reg[7:0] (hex) | Analog Cntrl [5:4] (binary) | Digital Input | Analog Voltage output | |
|---|---|---|---|---|---|
| | | | | General | 0-3V (approx) |
| -256 | 80H | 0x | 100H | Vdd/4 | 0.750V |
| -255 | 80H | 1x | 101H | Vdd/4+ 1 LSB | 0.753V |
| -254 | 81H | 0x | 102H | | 0.756V |
| -253 | 81H | 1x | 103H | | 0.759V |
| -252 | 82H | 0x | 104H | | 0.762V |
| | | | | | |
| -2 | FFH | 0x | 1FEH | | |
| -1 | FFH | 1x | 1FFH | Vdd/2- 1LSB | 1.497V |
| 0 | 00H | 0x | 200H | Vdd/2 | 1.500V |
| +1 | 00H | 1x | 201H | Vdd/2+ 1LSB | 1.503V |
| +2 | 01H | 0x | 202H | | |
| +3 | 01H | 1x | 203H | | |
| +4 | 02H | 0x | 204H | | |
| | | | | | |
| +254 | 7FH | 0x | 2FEH | | 2.244V |
| +255 | 7FH | 1x | 2FFH | 3Vdd/4-1 LSB | 2.247V |

The translation in Half-Scale mode is:
R2R[9] = dac[7] inverted
R2R[8:1] = dac[7:0]
R2R[0] = anCtl[5]

***DAC Power Control***

The DAC has no explicit power control. It is turned off (placed into lowest current mode) by loading the value 80H into the DAC hold register, and 0 into the LSB1 and LSB0 bits of the Analog Control Register (register EF.Bits[5:4]).

**Note: register EF.Bit7 ("-anCtl" must be "0" in the value being written to register EF, when writing EF.Bits[5:0].**

## Pulse Width Modulator (PWM) Analog Output

The PWM consists of circuitry to regulate the width of a pulse supplied to one of two outputs, PWM0 and PWM1, over a period of programmable duration. One or the other of the two outputs is held at ground and the other is driven with a pulse of programmable duration, giving "push-pull" drive. Both outputs have "low shoot-thru" transistors to reduce radiated EMI. Once programmed, the PWM produces outputs continuously until register values are changed. The PWM has both 8 and 10 bit modes. The PWM Control Register ('pwmCtl"; register D7) contains the PWM on/off control (Bit0), the sample period (Bits[3:2]), sample size selection controls (Bit5), and the two least-significant bits of the 10-bit output value (Bits[7:6]). The sample size defaults to 8 bits, with register D7.Bit5=0 ("tenBits"). A sample size of 10 bits is selected by setting "tenBits" =1. The PWM output impedance is approximately 11 Ohms. Of the standard speaker impedances available, an 8 ohm speaker will provide optimal volume when driven by the PWM.

The PWM contains two counters. The data value counter is programmed with the value programmed in the "pwmData" register (register D8) in 8-bit mode. In 10-bit mode the data value counter uses "pwmData" and appends Bits[7:6] of "pwmCtl" as the least significant two bits to create a 10 bit value. Output data always lags input by one PWM sample period. The sample period counter is fixed and counts to 128. The prescaler in the PWM control register (register D7.Bits[3:2]) determines the clock for both the data value counter and the sample period counter. The prescaler divides the 14.3 MHz clock by 4,6, or 7, resulting in a PWM frequency of 27.9 KHz, 18.6Khz and 15.97 KHz, respectively. The PWM restarts every sample period, at which time either PWM0 or PWM1 pulses high. The selected signal pulses high for a duration determined by the data value and then returns low. The non-selected signal remains low. The pulsed output selection is controlled by the sign of the data. When Bit 7 of the "pwmData" register is 0, PWM0 pulses high while PWM1 remains 0. When Bit 7 of the "pwmData" register is 1, PWM1 pulses high while PWM0 remains low. When the data value in "pwmData" is 0, both signals remain low. When the sample period count selected by programming Bits[3:2] of the "pwmCtl" register D7.Bit has been reached, the PWM restarts. The PWM hardware sample period and the software data value updating must be synchronized to avoid aliasing.
The following table shows the rates and pulse durations obtained for 8-bit mode ("tenBits" programmed to "0") SOFTWARE NOTE: "Full scale" output for all prescaler values is obtained by setting the data value to 7FH, so 8-bit signed data can be output at any of the three rates without amplitude adjustment.

PWM timing for "tenBits"=0

| Item | prescaler=4 | prescaler=6 | prescaler=7 |
|---|---|---|---|
| nsec/clock (period clock) | 280 | 420 | 490 |
| CLK1 clocks per period | 512 | 768 | 896 |
| nsec/clock (sample clock) | 280 | 420 | 490 |
| PWM frequency | 27.9 kHz | 18.6 kHz | 15.97 kHz |
| pulse for data=01 | 4 H / 508 L | 6 H / 762 L | 7 H / 889 L |
| pulse for data=7F | 508 H / 4L | 762 H / 6 L | 889 H / 7 L |

For 10-bit mode ("tenBits" programmed to "1"), the sample period counter counts a full 7-bits (128 counts), exactly as when TenBits is 0. The 14.3 MHz clock is divided by the prescaler value and supplied to the sample period counter. The data value counter is clocked by the 14.3 clock divided by 2 for prescaler values 6 or 7, and is clocked directly by the 14.3 MHz clock when the prescaler value is 4. The next table shows the rates and pulse durations obtained with tenBits set to 1. SOFTWARE NOTE: "Full scale" output is obtained with a different data value for each prescaler value. Only prescaler=4 supports a full 9-bit count (512), so true 10-bit signed data can be output only with prescaler=4. Otherwise the amplitudes must be adjusted to have maximum amplitude of 447 (prescaler=7) or 383 (prescaler=6). See "Additional considerations using the PWM for 10-bit Data" below.

PWM timing for tenBits=1

| Item | prescaler=4 | prescaler=6 | prescaler=7 |
|---|---|---|---|
| nsec/clock (period ctr) | 280 | 420 | 490 |
| CLK1 clocks per period | 512 | 768 | 896 |
| nsec/clock (data ctr) | 70 | 140 | 140 |
| PWM frequency | 27.9 kHz | 18.6 kHz | 15.97 kHz |
| pulse for data=001 | 1 H / 511 L | 2 H / 766 L | 2 H / 894 L |
| pulse for data=17F | 383 H / 129 L | 766 H / 2 L | 766 H / 130 L |
| pulse for data=1BF | 447 H / 65 L | -- n/a -- | 894 H / 2 L |
| pulse for data=1FF | 511 H / 1 L | -- n/a -- | -- n/a -- |

***Additional considerations using the PWM for 10-bit Data***

The 14.3 MHz CLK1 clock rate of the RSC-464 is not fast enough to provide PWM synchronization with 10-bit 8kHz or 9.3 kHz data. To understand this, consider a PWM rate of 8 kHZ (125 microsec). To output 10 bits (9 bits plus sign) during this interval, a source must provide 512 clocks, giving a source rate of 125000/512 = 244 nsec. The CLK1 period is 70 nsec, so the relationship between the source clock and CLK1 is 244/70 = 3.5, which is not an integer. So the source clock cannot be derived simply from CLK1.

The RSC-464 application developer should address this issue by using a "near-10-bit" resolution, as follows. The TenBits bit is *set* in the "pwmCtl" register, and the prescaler is programmed to 7 to produce a PWM frequency of 15.98 kHz (62.57 microseconds). During this interval there will be 62570/70 = 894 CLK1 clocks, or 894/2 (=447) data counter clocks. The number 447 thus represents the largest possible count that can be loaded into the data value counter. The range of allowable values is from –447 to +447. Any larger value would produce the same output of the PWM pulse "on" for the entire duration of the PWM period. Thus 447 represents "full scale" of the PWM. If all 10-bit data values are then scaled to a maximum of +/-447, the PWM will provide full-scale swing and (close-enough) synchronization at 8 kHz. The actual number of bits in the data is log2(447 – (-447)) = 9.8 bits. The developer must ensure that the value programmed in the data value counter must not exceed the range of –447 to +447. FluentChip™ provides PWM output utilities for speech and music that manage the PWM for the developer, if so desired. (See "FluentChip™ Technology Library Manual")

***PWM powerdown***

The PWM may be independently powered down by programming the register D7.Bit 0 to "0" ("pwmCtl" register, "pwm_on" bit). When the PWM is off, the PWM outputs PWM0 and PWM1 are in a high-Z state and pulled up by internal 15K resistors. The PWM must be explicitly turned off before setting "pdn" equal to 1 to achieve the lowest powerdown current.

## Comparator Unit

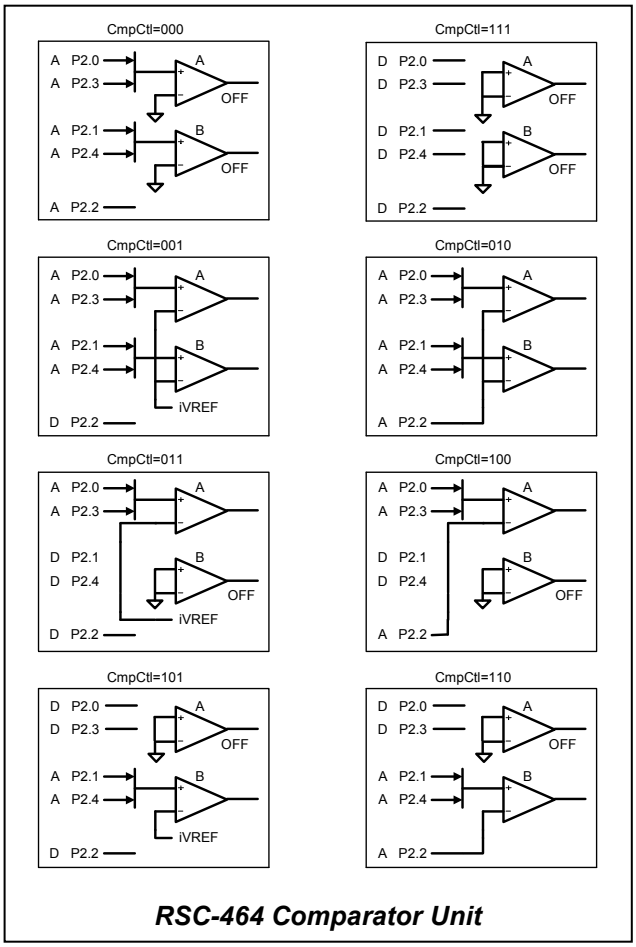
The Comparator Unit consists of 2 analog comparators designated "A" and "B", a programmable voltage reference, selection circuitry, and two registers – the Comparator Control register ("cmpCtl") and the Comparator Reference ("cmpRef"). Register "cmpCtl" configures the comparator unit and provides the digital comparator outputs. Bits [2:0] are used to select from one of eight comparator configurations, in which some or all of P2.0-P2.4 may be analog or digital inputs. (See "RSC-464 Comparator Unit" figure; "A" denotes analog input and "D" denotes digital input) Bits [3:0] are read-write.

Register "cmpRef" controls the Comparator Reference Voltage. The unit can provide level information under software control about 4 external analog signals. All external signals connected to the comparator inputs must be between Vss and Vdd.

Each comparator has two analog inputs, designated "+" and "-", and one digital output. When the analog voltage on the "+" input is greater than the analog voltage on the "-" input, the digital output is a high level. This is indicated by a "1" in the "cmpCtl" register (register D4) Bits 7 & 6 for Comparators A and B, respectively. When the analog voltage on the "+" input is less than the analog voltage on the "-" input, the digital output is a low level. This is indicated by a "0" in the "cmpCtl" register (register D4) Bits 7 & 6 for Comparators A and B, respectively. Bits 7 and 6 are the comparator outputs and are "Read-Only" by the processor.

Each comparator can be separately enabled or disabled. When a comparator is disabled, both inputs are isolated from any circuitry common to both comparators, the inputs are grounded, and the comparator power is turned off.




***RSC-464 Comparator Unit***

### *Comparator Multiplexing*

Each comparator "+" input has an analog multiplexer that selects between one of two external signals. When Bit3 of "cmpCtl" is programmed to "0", comparator input A+ is multiplexed to P2.0 and input B+ is multiplexed to P2.1. When Bit3 of "cmpCtl" is programmed to "1", comparator input A+ is multiplexed to P2.3 and input B+ is multiplexed to P2.4. The "-" inputs of both comparators are connected together. This common "-" input can be multiplexed to either an external comparator reference signal input through P2.2, or the Comparator Reference Voltage (CRV).

### *Comparator Reference Voltage*

The internal Comparator Reference Voltage (CRV) is derived from a multi-tap resistive divider and a 4-bit analog multiplexer. Register "cmpRef" controls the Comparator Reference Voltage. The power for the Comparator Reference Voltage is provided by unregulated Vdd. This means that the CRV will track external voltages referenced from the system supply, giving consistent comparisons as the system supply drops. Power to the CRV is gated by decoding the comparator configuration. The voltage select value in "cmpRef" Bits[3:0] selects one of 16 outputs of an analog multiplexer connected to 16 equally spaced taps. The Comparator Reference Voltage covers the range from 0.15*Vdd to 0.90*Vdd in steps of 0.05*Vdd and is given by 0.15*Vdd + (D3[3:0]/20)*Vdd.

In some configurations the Comparator Control register can be set up once and simply read thereafter. In many configurations it will be necessary to switch the input multiplexers and/or re-program the reference voltage repeatedly. These multiplexing and selection operations will have settling times of approximately 10 microseconds.

When the "pdn" bit is set for Idle or Sleep mode the entire comparator unit is powered down, but the contents of the "cmpCtl" and "cmpRef" registers are preserved. When the RSC-464 wakes up the comparators resume normal operation.

# Instruction Set Opcodes and Timing Details

The RSC-464 instruction set has 60 instructions comprising 13 move, 7 rotate/shift, 11 jump/branch, 13 register arithmetic, 9 immediate arithmetic, and 7 miscellaneous instructions. All instructions are 3 bytes or fewer, and no instruction requires more than 10 clock cycles (plus wait states) to execute. The column "Cycles" indicates the number of clock cycles required for each instruction when operating with zero wait states. Wait states may be added to lengthen all accesses to external addresses or to the internal ROM (but not internal SRAM). The column "+Cycles/Waitstate" shows the number of additional cycles added for each additional wait state. Opcodes are in HEX.

## MOVE Group Instructions

Register-indirect instructions accessing code (*MOVC*), data (*MOVX*), technology (*MOVY*) or register (*MOV*) space locations use an 8-bit operand ("@source" or "@dest") to designate an SRAM register pointer to the 16-bit target address. The "source" or "dest" indirect pointer register must be at an even address unless it is a 8-bit pointer (indirect *MOV*). The LOW byte of the target address is contained at the pointer address, and the HIGH byte of the target address is contained at the pointer address+1. Unless the flags register is the destination, the carry, sign, and zero flags are not affected by *MOV* instructions.

| Instruction | Opcode | Operand 1 | Operand 2 | Description | Bytes | Cycles | +Cycles/Waitstate |
|---|---|---|---|---|---|---|---|
| MOV | 10 | dest | Source | register to register | 3 | 5 | 3 |
| MOV | 11 | @dest | Source | register to register-indirect | 3 | 5 | 3 |
| MOV | 12 | dest | @source | register-indirect to register | 3 | 6 | 3 |
| MOV | 13 | dest | #immed | immediate data to register | 3 | 4 | 3 |
| MOVC | 14 | dest | @source | code space to register | 3 | 7 | 4 |
| MOVC | 15 | @dest | Source | register to code space | 3 | 8 | 4 |
| MOVX | 16 | dest | @source | data space to register | 3 | 7 | 4* |
| MOVX | 17 | @dest | Source | register to data space | 3 | 8 | 4* |
| POP | 18 | dest | @++source | register to register data stack pop (source pre-incremented) | 3 | 10 | 3 |
| PUSH | 19 | @dest-- | Source | register to register data stack push (dest post-decremented) | 3 | 9 | 3 |
| MOVY | 1A | dest | @source | RAMY to register, indirect | 3 | 7 | 3 |
| MOVY | 1B | @dest | source | Register to RAMY, indirect | 3 | 7 | 3 |
| MOVD | 1C | dest_pair | source_pair | register to register, direct, 16-bit MOV | 3 | 7 | 3 |

* If register D6.Bit 5=1 (movX_4ws) and external read/write memory is selected by setting the "rw" bit (register D2.Bit4), MOVX instructions have four additional wait states.

## ROTATE Group Instructions

Rotate group instructions apply only directly to register space SRAM locations. The carry flag is affected by these instructions, but the sign and zero flags are unaffected.

| Instruction | Opcode | Operand 1 | Operand 2 | Description | Bytes | Cycles | +Cycles/Waitstate |
|---|---|---|---|---|---|---|---|
| RL | 30 | dest | - | rotate left, c set from b7 | 2 | 5 | 2 |
| RR | 31 | dest | - | rotate right, c set from b0 | 2 | 5 | 2 |
| RLC | 32 | dest | - | rotate left through carry | 2 | 5 | 2 |
| RRC | 33 | dest | - | rotate right through carry | 2 | 5 | 2 |
| SHL | 34 | dest | - | shift left, c set from b7, b0=0 | 2 | 5 | 2 |
| SHR | 35 | dest | - | shift right, c set from b0, b7=0 | 2 | 5 | 2 |
| SAR | 36 | dest | - | shift right arithmetic, c set from b0, b7 duplicated | 2 | 5 | 2 |

## BRANCH Group Instructions

The branch instructions use direct address values rather than offsets to define the target address of the branch. This implies that binary code containing branches is not relocatable. However, object code produced by the RSC-464 assembler contains address references that are resolved at link time, so .OBJ modules *are* relocatable. The indirect jump instruction uses an 8-bit operand ("@dest") to designate an SRAM register pointer to the 16-bit target address. The "dest" pointer register must be at an even address. The LOW byte of the target address is contained at the pointer address, and the HIGH byte of the target address is contained at the pointer address+1.

| Instruction | Opcode | Operand 1 | Operand 2 | Description | Bytes | Cycles | +Cycles/Waitstate |
|---|---|---|---|---|---|---|---|
| JC | 20 | dest low | dest high | jump on carry = 1 | 3 | 3 | 3 |
| JNC | 21 | dest low | dest high | jump on carry = 0 | 3 | 3 | 3 |
| JZ | 22 | dest low | dest high | jump on zflag = 1 | 3 | 3 | 3 |
| JNZ | 23 | dest low | dest high | jump on zflag = 0 | 3 | 3 | 3 |
| JS | 24 | dest low | dest high | jump on sflag = 1 | 3 | 3 | 3 |
| JNS | 25 | dest low | dest high | jump on sflag = 0 | 3 | 3 | 3 |
| JMP | 26 | dest low | dest high | jump unconditional | 3 | 3 | 3 |
| CALL | 27 | dest low | dest high | direct subroutine call | 3 | 3 | 3 |
| RET | 28 | - | - | return from call | 1 | 2 | 1 |
| IRET | 29 | - | - | return from interrupt | 1 | 2 | 1 |
| JMPR | 2A | @dest | - | jump indirect | 2 | 4 | 2 |

## ARITHMETIC/LOGICAL Group Instructions

Arithmetic and logical group instructions apply only to Register Space SRAM locations. The results of the instruction are always written directly to the SRAM "dest" register. The exceptions are TM and CP instructions, which do not write the result to the "dest" register and only update the flags register based on the operation's outcome. All but the INCrement and DECrement instructions have both register source and immediate source forms.

In each of the following instructions the sign and zero flags are updated based on the result of the operation. The carry flag is updated by the arithmetic operations (ADD, ADC, SUB, SUBC, CP, INC, DEC) but it is *not* affected by the logical operations (AND, TM, OR, XOR). Note: the carry is set **high** by SUB, CP, SUBC and DEC when a borrow is generated.

| Instruction | Opcode | Operand 1 | Operand 2 | Description | Bytes | Cycles | +Cycles/Waitstate |
|---|---|---|---|---|---|---|---|
| AND | 40 | dest | source | logical and | 3 | 6 | 3 |
| TM | 41 | dest | source | like AND, destination register unchanged | 3 | 6 | 3 |
| OR | 42 | dest | source | logical or | 3 | 6 | 3 |
| XOR | 43 | dest | source | exclusive or | 3 | 6 | 3 |
| SUB | 44 | dest | source | subtract | 3 | 6 | 3 |
| CP | 45 | dest | source | like SUB, destination register unchanged | 3 | 6 | 3 |
| SUBC | 46 | dest | source | subtract w/carry | 3 | 6 | 3 |
| ADD | 47 | dest | source | add | 3 | 6 | 3 |
| ADC | 48 | dest | source | add w/carry | 3 | 6 | 3 |
| INC | 49 | dest | - | increment | 2 | 5 | 2 |
| DEC | 4A | dest | - | decrement | 2 | 5 | 2 |
| AND | 50 | dest | #immed | logical and | 3 | 5 | 3 |
| TM | 51 | dest | #immed | like AND, destination register unchanged | 3 | 5 | 3 |
| OR | 52 | dest | #immed | logical or | 3 | 5 | 3 |
| XOR | 53 | dest | #immed | exclusive or | 3 | 5 | 3 |
| SUB | 54 | dest | #immed | subtract | 3 | 5 | 3 |
| CP | 55 | dest | #immed | like SUB, destination register unchanged | 3 | 5 | 3 |
| SUBC | 56 | dest | #immed | subtract w/carry | 3 | 5 | 3 |
| ADD | 57 | dest | #immed | add | 3 | 5 | 3 |
| ADC | 58 | dest | #immed | add w/carry | 3 | 5 | 3 |
| INCD | 69 | dest_pair & source_pair | - | register pair 16-bit increment | 2 | 8 | 2 |
| CPD | 66 | dest_pair | source_pair | 16-bit compare | 3 | 10 | 3 |

## MISCELLANEOUS Group Instructions

| Instruction | Opcode | Operand 1 | Operand 2 | Description | Bytes | Cycles | +Cycles/Waitstate |
|---|---|---|---|---|---|---|---|
| NOP | 00 | - | - | no operation | 1 | 2 | 1 |
| CLC | 01 | - | - | clear carry | 1 | 2 | 1 |
| STC | 02 | - | - | set carry | 1 | 2 | 1 |
| CMC | 03 | - | - | complement carry | 1 | 2 | 1 |
| CLI | 04 | - | - | disable interrupts | 1 | 2 | 1 |
| STI | 05 | - | - | enable interrupts | 1 | 2 | 1 |
| WDC | 06 | - | - | enable/restart Watchdog timer | 1 | 2 | 1 |

# Special Functions Registers (SFRs) Summary

| Address | R/W | Name | Reset | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| FF | R/W | flags ***** | 0000 0000 | carry | zero | sign | trap | stkoflo | stkfull | --- | gie |
| FE | R/W | irq * | 0000 0000 | MTtimer | p0.2 | block | timer3 | p0.0 | endmark | timer2 | timer1 |
| FD | R/W | imr **** | 0000 0000 | MTtimer | p0.2 | block | timer3 | p0.0 | endmark | timer2 | timer1 |
| FC | R/W | bank | 1110 0000 | ws2 | ws1 | ws0 | (bank4) | bank3 | bank2 | bank1 | bank0 |
| FB | W | RESERVED | | | | | | | | | |
| | R | sysStat | 0000 0000 | 0 | 1 | wd_timed | wd_on | 0 | 0 | fastClk | 0 |
| FA | R/W | dac | 0000 0000 | dh7 | dh6 | dh5 | dh4 | dh3 | dh2 | dh1 | dh0 |
| F9 | R/W | RESERVED | | | | | | | | | |
| F8 | R/W | RESERVED | | | | | | | | | |
| F7 | R/W | stkData | 0000 0000 | stdk7 | stkd6 | stkd5 | stkd4 | stkd3 | stkd2 | stkd1 | stkd0 |
| F6 | R/W | stkNdx | 0000 0000 | 0 | 0 | stkind5 | stkind4 | stkind3 | stkind2 | stkind1 | stkind0 |
| F5 | W | RESERVED | | | | | | | | | |
| | R | adcSampleHi | 0000 0000 | adc15 | adc14 | adc13 | adc12 | adc11 | adc10 | adc09 | adc08 |
| F4 | W | RESERVED | | | | | | | | | |
| | R | adcSampleLo | 0000 0000 | adc07 | adc06 | adc05 | adc04 | adc03 | adc02 | adc01 | adc00 |
| F3 | R/W | RESERVED | | | | | | | | | |
| F2 | R/W | RESERVED | | | | | | | | | |
| F1 | W | RESERVED | | | | | | | | | |
| | R | RESERVED | | | | | | | | | |
| F0 | R/W | RESERVED | | | | | | | | | |
| EF | W | anCtl *** | 0000 0000 | -anctlen | 0 | lsb1 | lsb0 | d2a_half | rc_osc2 | 0 | afe_on |
| | R | | 0000 0000 | -anctlen | 0 | lsb1 | lsb0 | d2a_half | rc_osc2 | 0 | afe_on |
| EE | W | t2v ** | 0000 0000 | x | x | x | x | x | x | x | x |
| | R | 0000 0000 | | t2v7 | t2v6 | t2v5 | t2v4 | t2v3 | t2v2 | t2v1 | t2v0 |
| ED | R/W | t2r | 0000 0000 | t2r7 | t2r6 | t2r5 | t2r4 | t2r3 | y2r2 | t2r1 | t2r0 |
| EC | W | t1v ** | 0000 0000 | x | x | x | x | x | x | x | x |
| | R | 0000 0000 | | t1v7 | t1v6 | t1v5 | t1v4 | t1v3 | t1v2 | t1v1 | t1v0 |
| EB | R/W | t1r | 0000 0000 | t1r7 | t1r6 | t1r5 | t1r4 | t1r3 | t1r2 | t1r1 | t1r0 |
| EA | R/W | wake1 | 0000 0000 | w1.7 | w1.6 | w1.5 | w1.4 | w1.3 | w1.2 | w1.1 | w1.0 |
| E9 | R/W | wake0 | 0000 0000 | w0.7 | w0.6 | w0.5 | w0.4 | w0.3 | w0.2 | w0.1 | w0.0 |
| E8 | R/W | ckCtl **** | 0000 1000 | pdn | t2wake | fclk_on | clk_div1 | clk_div0 | slow_pclk | osc2_on | osc1_off |
| E7 | R/W | p0CtlB | 0000 0000 | ctlb0.7 | ctlb0.6 | ctlb0.5 | ctlb0.4 | ctlb0.3 | ctlb0.2 | ctlb0.1 | ctlb0.0 |
| E6 | R/W | p0CtlA | 0000 0000 | ctla0.7 | ctla0.6 | ctla0.5 | ctla0.4 | ctla0.3 | ctla0.2 | ctla0.1 | ctla0.0 |
| E5 | R | p0In | xxxx xxxx | pin0.7 | pin0.6 | pin0.5 | pin0.4 | pin0.3 | pin0.2 | pin0.1 | pin0.0 |
| E4 | R/W | p0Out | 0000 0000 | pout0.7 | pout0.6 | pout0.5 | pout0.4 | pout0.3 | pout0.2 | pout0.1 | pout0.0 |
| E3 | | RESERVED | | | | | | | | | |
| E2 | | RESERVED | | | | | | | | | |
| E1 | | RESERVED | | | | | | | | | |
| E0 | | RESERVED | | | | | | | | | |
| DF | R/W | p2CtlB | 0000 0000 | ctlb2.7 | ctlb2.6 | ctlb2.5 | ctlb2.4 | ctlb2.3 | ctlb2.2 | ctlb2.1 | ctlb2.0 |

| Address | R/W | Name | Reset | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| DE | R/W | p2CtlA | 0000 0000 | ctla2.7 | ctla2.6 | ctla2.5 | ctla2.4 | ctla2.3 | ctla2.2 | ctla2.1 | ctla2.0 |
| DD | R | p2In | xxxx xxxx | pin2.7 | pin2.6 | pin2.5 | pin2.4 | pin2.3 | pin2.2 | pin2.1 | pin2.0 |
| DC | R/W | p2Out | 0000 0000 | pout2.7 | pout2.6 | pout2.5 | pout2.4 | pout2.3 | pout2.2 | pout2.1 | pout2.0 |
| DB | W | t3v ** | 0000 0000 | x | x | x | x | x | x | x | x |
| | R | | 0000 0000 | t3v7 | t3v6 | t3v5 | t3v4 | t3v3 | t3v2 | t3v1 | t3v0 |
| DA | R/W | t3r | 0000 0000 | t3r7 | t3r6 | t3r5 | t3r4 | t3r3 | t3r2 | t3r1 | t3r0 |
| D9 | W | t3Ctl | 0000 0000 | t3_on | polarity | p0.1_src | t3_gated | t3_ps3 | t3_ps2 | t3_ps1 | t3_ps0 |
| D8 | R/W | pwmData | 0000 0000 | pwmd09 | pwmd08 | pwmd07 | pwmd06 | pwmd05 | pwmd04 | pwmd03 | pwmd02 |
| D7 | R/W | pwmCtl | 0000 0000 | pwmd01 | pwmd00 | tenBits | 0 | period1 | period0 | 0 | pwm_on |
| D6 | R/W | clkExt **** | 0000 0000 | rom_0Ws | MTclk_on | movx_4ws | L1clk_on | t1_ps3 | t1_ps2 | t1_ps1 | t1_ps0 |
| D5 | R/W | sysCtl | 0000 0000 | wd_ps1 | wd_ps0 | brnout_on | afe_g1 | afe_g0 | 0 | p02Edge | p00Edge |
| D4 | W | cmpCtl | 1100 0000 | 1 | 1 | 0 | 0 | mux_sel | ccs2 | ccs1 | ccs0 |
| | R | | 1100 0000 | compA+ | compB+ | 0 | 0 | mux_sel | ccs2 | ccs1 | ccs0 |
| D3 | R/W | cmpRef | 0000 0000 | 0 | 0 | 0 | 0 | crv03 | crv02 | crv01 | crv00 |
| D2 | R/W | extAdd | 0000 0000 | 0 | 0 | cb1 | rw | eda19 | eda18 | eda17 | eda16 |
| D1 | R/W | RESERVED | | | | | | | | | |
| D0 | R/W | RESERVED | | | | | | | | | |
| CF | R/W | flagsHold ***** | 0000 0000 | carry | zero | sign | trap | 0 | 0 | 0 | gie |
| CE | W | awcCtl | 0000 0000 | pwrl | 0 | thrh2 | thrh1 | thrh0 | thrl2 | thrl1 | thrl0 |

| Address | R/W | Name | Reset | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | R | | 0000 0000 | pwrl | detect | thrh2 | thrh1 | thrh0 | thrl2 | thrl1 | thrl0 |
| | | | | | | | | | | | |
| CD | | RESERVED | | | | | | | | | |
| | | RESERVED | | | | | | | | | |
| CC | | RESERVED | | | | | | | | | |
| CB | | RESERVED | | | | | | | | | |
| CA | | RESERVED | | | | | | | | | |
| C9 | | RESERVED | | | | | | | | | |
| C8 | | RESERVED | | | | | | | | | |
| | | RESERVED | | | | | | | | | |
| C7 | | RESERVED | | | | | | | | | |
| C6 | | RESERVED | | | | | | | | | |
| C5 | | RESERVED | | | | | | | | | |
| C4 | | RESERVED | | | | | | | | | |
| C3 | | RESERVED | | | | | | | | | |
| | | RESERVED | | | | | | | | | |
| C2 | | RESERVED | | | | | | | | | |
| | | RESERVED | | | | | | | | | |
| C1 | | RESERVED | | | | | | | | | |
| | | RESERVED | | | | | | | | | |
| C0 | | RESERVED | | | | | | | | | |
| | | RESERVED | | | | | | | | | |

Reset: "x" = unknown/don't care, '-' = not implemented

***** Only "0" can be written to "irq" bits. "1" is a "nop" for the bit to which it is written. When using FluentChip™ technology, always write "1" to "block" and "endmark" in the "irq" register to avoid conflicting with technology code control of these bits.

****** Write value is ignored and reload register value is written instead.

*** -anctlen (Bit7) of values written to the "anCtl" register must be "0" to enable writing the other bits in the value to "anCtl".

**** When using FluentChip™ technology, "fclk_on", "L1clk_on", and "block" and "endmark" in the "imr" register should be left at the values programmed by the technology code. A read-modify-write action should be used to modify the registers to avoid changing these bits.

***** "trap" must always be written as "0" in the "flags" and "flagsHold" registers

## DC Characteristics

Operating Conditions ($T_O$ = 0°C to +70°C, $V_{DD}$ = 2.4V – 3.6V)

| SYMBOL | PARAMETER | MIN | TYP | MAX | UNITS | TEST CONDITIONS |
|---|---|---|---|---|---|---|
| $V_{IL}$ | Input Low Voltage | -0.1 | | 0.75 | V | |
| $V_{IH}$ | Input High Voltage | 0.8*Vdd | | Vdd+0.3 | V | |
| $I_{IL}$ | Input Leakage Current | | <1 | 10 | µA | $V_{ss}<V_{pin}<V_{dd}$ |
| $I_{ACT}$ | Supply Current, Active | | 10 | | mA | Hi-Z Outputs, Vdd=3V |
| $I_{ACT}$ | Supply Current, Active | | | 20 | mA | Hi-Z Outputs, Vdd=3.6V |
| $I_{IDLE}$ | Supply Current, Idle without Audio Wakeup | | 4 | 7 | µA | Hi-Z Outputs |
| $I_{AW}$ | Supply Current, Idle with Audio Wakeup (1) | | 40 | 70 | uA | Hi-Z Output |
| $I_{SLEEP}$ | Supply Current, Sleep | | 1 | 4 | µA | Hi-Z Outputs |
| $R_{PU}$ | Pull-up resistance | | | | | |
| | P0.0-P0.7, P2.0- P2.7 | | 10, 200, Hi-Z | | kΩ | Software selectable |
| | PLLEN, -RESET | | 100 | | kΩ | Fixed |
| | PWM0, PWM1 | | 5 | | kΩ | Fixed |
| $I_{OL}$ | Output Low Current | | | | | |
| | PDN | 4 | | | mA | $V_{OL}$ = 0.5V, $V_{DD}$ = 2.4V |
| | P0.0-P0.7, P2.0-P2.7 | 8 | | | mA | $V_{OL}$ = 0.5V, $V_{DD}$ = 2.4V |
| | PWM0, PWM1 | | 180 | | mA | $V_{OL}$ = 0.8V, $V_{DD}$ = 3.3V |
| $I_{OH}$ | Output High Current | | | | | |
| | PDN | -2.5 | | | mA | $V_{OH}$ = 1.8V, $V_{DD}$ = 2.4V |
| | P0.0-P0.7, P2.0-P2.7 | -5 | | | mA | $V_{OH}$ = 1.8V, $V_{DD}$ = 2.4V |
| | PWM0, PWM1 | | -80 | | mA | $V_{OH}$ = 2.5V, $V_{DD}$ = 3.3V |

NOTES

1. Audio Wakeup also requires a microphone to be active. The microphone current will vary depending on the design, but a typical value is approximately 100uA. System level consumption will be the sum of the microphone current and $I_{AW}$ .

## Absolute Maximum Ratings

| | |
|---|---|
| Any pin to GND: | -0.1V to +4.0V |
| Storage temperature: | -65°C to +150°C |
| Operating temperature: | -40°C to +85°C |
| Soldering temperature: | 260°C for 10 sec |
| Power dissipation: | 1 W |

**WARNING**:

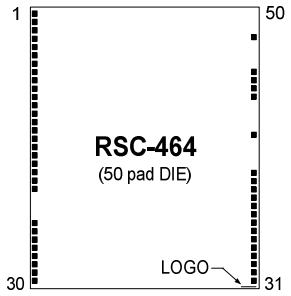
Stressing the RSC-464 beyond the "Absolute Maximum Ratings" may cause permanent damage. These are stress ratings only.
Operation beyond the "Operating Conditions" is not recommended and extended exposure beyond the "Operating Conditions" may affect device reliability.

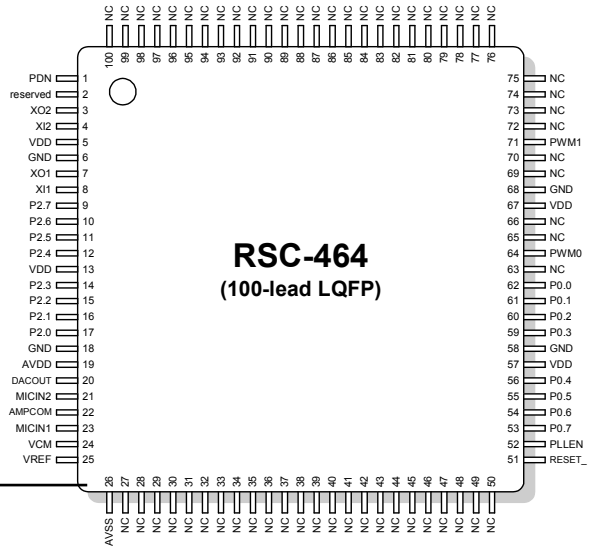
## Package Options

The RSC-464 can be purchased in a 100-pin LQFP package or in unpackaged die. When using an in circuit emulator (ICE) on dice applications, a COB bonding pad ring equivalent to a 100-pin LQFP footprint is advised for easy ICE adapter attachment.

**DIE**




**100-pin LQFP**



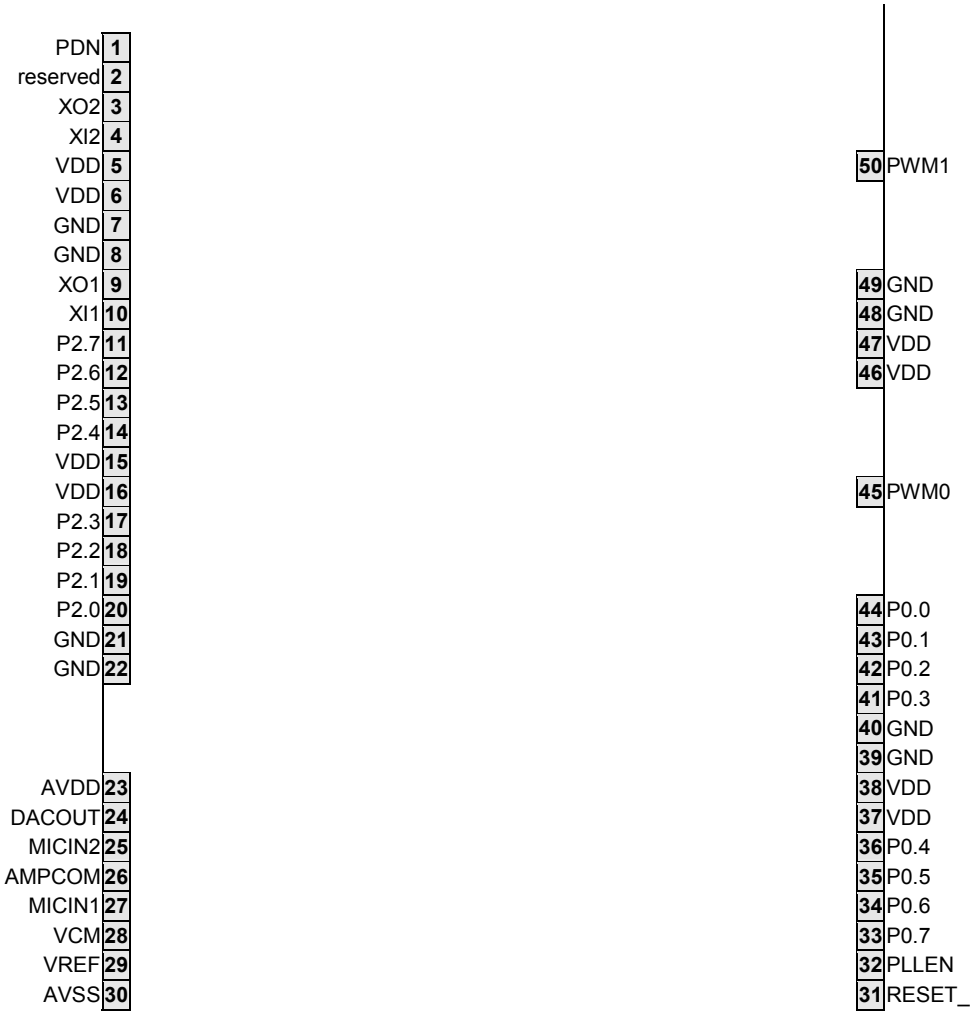

| DIE Pad # | ---------- | 100 LQFP Pin # | Pin Name | Description | Signal Type |
|---|---|---|---|---|---|
| 1 | | 1 | **PDN** | Power Down (active high when powered down) | Output |
| 2 | | 2 | **reserved** | DO NOT USE | DO NOT USE |
| 3 | | 3 | **XO2** | Oscillator 2 output | Output |
| 4 | | 4 | **XI2** | Oscillator 2 input | Input |
| 5 | | 5 | **VDD** | Supply Voltage | PWR |
| 6 | | 5 | **VDD** | Supply Voltage | PWR |
| 7 | | 6 | **GND** | Ground | GND |
| 8 | | 6 | **GND** | Ground | GND |
| 9 | | 7 | **XO1** | Oscillator 1 output | Output |
| 10 | | 8 | **XI1** | Oscillator 1 input | Input |
| 11 | | 9 | **P2.7** | General Purpose I/O that can act as a “wake-up” input | I/O, 10k or 200k pull-up resistor; high-Z |
| 12 | | 10 | **P2.6** | General Purpose I/O that can act as a “wake-up” input | I/O, 10k or 200k pull-up resistor; high-Z |
| 13 | | 11 | **P2.5** | General Purpose I/O that can act as a “wake-up” input | I/O, 10k or 200k pull-up resistor; high-Z |
| 14 | | 12 | **P2.4** | General Purpose I/O or comparator input | I/O, 10k or 200k pull-up resistor; high-Z |
| 15 | | 13 | **VDD** | Supply Voltage | PWR |
| 16 | | 13 | **VDD** | Supply Voltage | PWR |
| 17 | | 14 | **P2.3** | General Purpose I/O or comparator input | I/O, 10k or 200k pull-up resistor; high-Z |
| 18 | | 15 | **P2.2** | General Purpose I/O or comparator reference | I/O, 10k or 200k pull-up resistor; high-Z |
| 19 | | 16 | **P2.1** | General Purpose I/O or comparator input | I/O, 10k or 200k pull-up resistor; high-Z |
| 20 | | 17 | **P2.0** | General Purpose I/O or comparator input | I/O, 10k or 200k pull-up resistor; high-Z |
| 21 | | 18 | **GND** | Ground | GND |
| 22 | | 18 | **GND** | Ground | GND |
| 23 | | 19 | **AVDD** | Analog Supply Voltage | Analog PWR |
| 24 | | 20 | **DACOUT** | DAC output | Analog out |
| 25 | | 21 | **MICIN2** | Microphone input for audio wakeup | Analog IN |
| 26 | | 22 | **AMPCOM** | Amplifier input common | Analog IN |
| 27 | | 23 | **MICIN1** | Microphone input | Analog IN |
| 28 | | 24 | **VCM** | Common mode reference | Analog |
| 29 | | 25 | **VREF** | Voltage reference | Analog OUT |
| 30 | | 26 | **AVSS** | Analog ground | Analog GND |
| - | | 27 | **NC** | Not connected | |
| - | | 28 | **NC** | Not connected | |
| - | | 29 | **NC** | Not connected | |
| - | | 30 | **NC** | Not connected | |

| DIE Pad # | ---------- | 100 LQFP Pin # | Pin Name | Description | Signal Type |
|---|---|---|---|---|---|
| - | | 31 | NC | Not connected | |
| - | | 32 | NC | Not connected | |
| - | | 33 | NC | Not connected | |
| - | | 34 | NC | Not connected | |
| - | | 35 | NC | Not connected | |
| - | | 36 | NC | Not connected | |
| - | | 37 | NC | Not connected | |
| - | | 38 | NC | Not connected | |
| - | | 39 | NC | Not connected | |
| - | | 40 | NC | Not connected | |
| - | | 41 | NC | Not connected | |
| - | | 42 | NC | Not connected | |
| - | | 43 | NC | Not connected | |
| - | | 44 | NC | Not connected | |
| - | | 45 | NC | Not connected | |
| - | | 46 | NC | Not connected | |
| - | | 47 | NC | Not connected | |
| - | | 48 | NC | Not connected | |
| - | | 49 | NC | Not connected | |
| - | | 50 | NC | Not connected | |
| 31 | | 51 | -RESET | Reset (active low) | Input, 100k pull-up resistor |
| 32 | | 52 | PLLEN | PLL Enable | Input, 100k pull-up resistor |
| 33 | | 53 | P0.7 | General Purpose I/O that can act as a “wake-up” input | I/O, 10k or 200k pull-up resistor; high-Z |
| 34 | | 54 | P0.6 | General Purpose I/O that can act as a “wake-up” input | I/O, 10k or 200k pull-up resistor; high-Z |
| 35 | | 55 | P0.5 | General Purpose I/O that can act as a “wake-up” input | I/O, 10k or 200k pull-up resistor; high-Z |
| 36 | | 56 | P0.4 | General Purpose I/O that can act as a “wake-up” input | I/O, 10k or 200k pull-up resistor; high-Z |
| 37 | | 57 | VDD | Supply Voltage | PWR |
| 38 | | 57 | VDD | Supply Voltage | PWR |
| 39 | | 58 | GND | Ground | GND |
| 40 | | 58 | GND | Ground | GND |
| 41 | | 59 | P0.3 | General Purpose I/O that can act as a “wake-up” input | I/O, 10k or 200k pull-up resistor; high-Z |
| 42 | | 60 | P0.2 | General Purpose I/O that can act as a “wake-up” input | I/O, 10k or 200k pull-up resistor; high-Z |
| 43 | | 61 | P0.1 | General Purpose I/O that can act as a “wake-up” input | I/O, 10k or 200k pull-up resistor; high-Z |
| 44 | | 62 | P0.0 | General Purpose I/O that can act as a “wake-up” input | I/O, 10k or 200k pull-up resistor; high-Z |
| - | | 63 | NC | Not connected | |
| 45 | | 64 | PWM0 | Pulse Width Modulator Output 0 | Output; 5k pull-up resistor; high-Z |
| - | | 65 | NC | Not connected | |
| - | | 66 | NC | Not connected | |
| 46 | | 67 | VDD | Supply Voltage | PWR |
| 47 | | 67 | VDD | Supply Voltage | PWR |
| 48 | | 68 | GND | Ground | GND |
| 49 | | 68 | GND | Ground | GND |
| - | | 69 | NC | Not connected | |
| - | | 70 | NC | Not connected | |
| 50 | | 71 | PWM1 | Pulse Width Modulator Output 1 | Output; 5k pull-up resistor; high-Z |
| - | | 72 | NC | Not connected | |
| - | | 73 | NC | Not connected | |
| - | | 74 | NC | Not connected | |
| - | | 75 | NC | Not connected | |
| - | | 76 | NC | Not connected | |
| - | | 77 | NC | Not connected | |
| - | | 78 | NC | Not connected | |
| - | | 79 | NC | Not connected | |
| - | | 80 | NC | Not connected | |
| - | | 81 | NC | Not connected | |
| - | | 82 | NC | Not connected | |
| - | | 83 | NC | Not connected | |
| - | | 84 | NC | Not connected | |
| - | | 85 | NC | Not connected | |
| - | | 86 | NC | Not connected | |
| - | | 87 | NC | Not connected | |
| - | | 88 | NC | Not connected | |
| - | | 89 | NC | Not connected | |
| - | | 90 | NC | Not connected | |
| - | | 91 | NC | Not connected | |
| - | | 92 | NC | Not connected | |
| - | | 93 | NC | Not connected | |

| DIE Pad # | ---------- | 100 LQFP Pin # | Pin Name | Description | Signal Type |
|---|---|---|---|---|---|
| - | | 94 | NC | Not connected | |
| - | | 95 | NC | Not connected | |
| - | | 96 | NC | Not connected | |
| - | | 97 | NC | Not connected | |
| - | | 98 | NC | Not connected | |
| - | | 99 | NC | Not connected | |
| - | | 100 | NC | Not connected | |

## Die Pad Ring



# RSC-464 Die Bonding Pad Locations

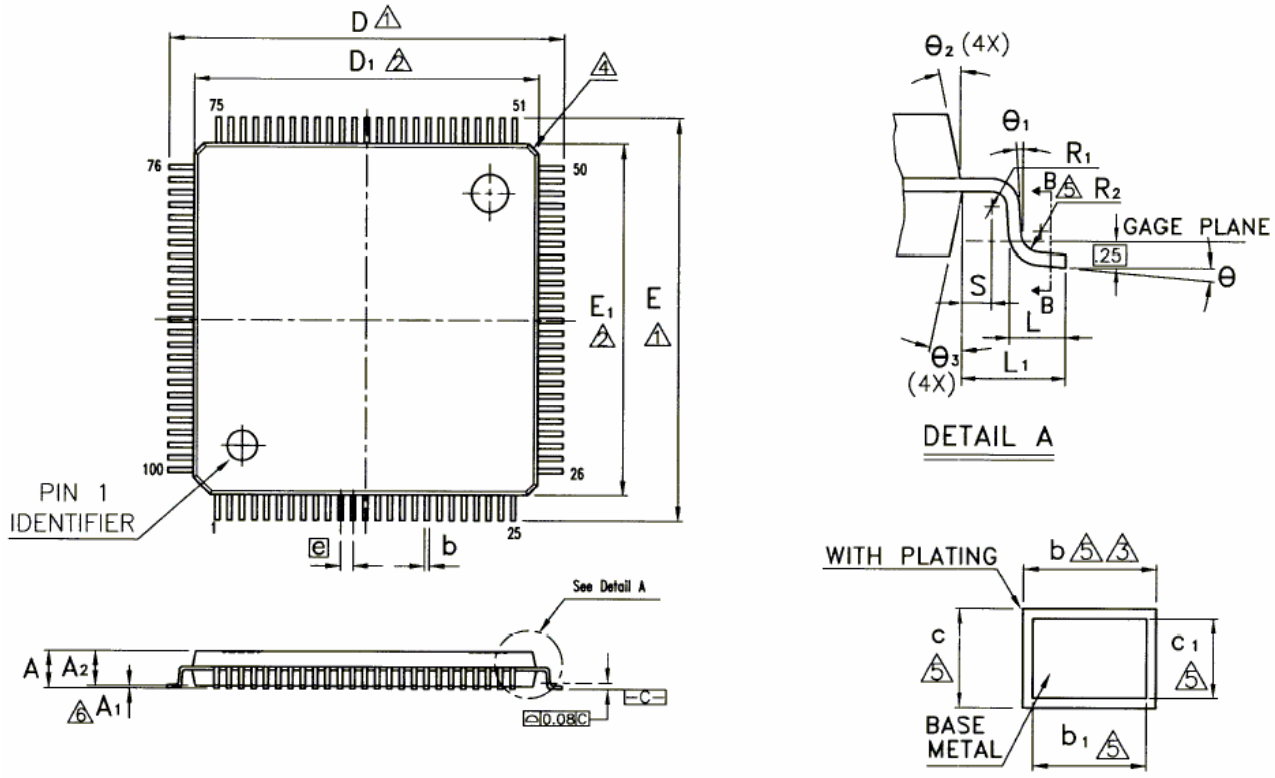
| PAD # | PADNAME | X (um) | Y (um) | PAD # | PADNAME | X (um) | Y (um) |
|---|---|---|---|---|---|---|---|
| ***1*** | PDN | 55 | 3330 | ***30*** | AVSS | 55 | 98 |
| ***2*** | reserved | 55 | 3228 | ***31*** | -RESET | 2750 | 96 |
| ***3*** | XO2 | 55 | 3127 | ***32*** | PLLEN | 2750 | 196 |
| ***4*** | XI2 | 55 | 3032 | ***33*** | P0.7 | 2750 | 296 |
| ***5*** | VDD | 55 | 2930 | ***34*** | P0.6 | 2750 | 396 |
| ***6*** | VDD | 55 | 2828 | ***35*** | P0.5 | 2750 | 497 |
| ***7*** | GND | 55 | 2726 | ***36*** | P0.4 | 2750 | 597 |
| ***8*** | GND | 55 | 2624 | ***37*** | VDD | 2750 | 697 |
| ***9*** | XO1 | 55 | 2523 | ***38*** | VDD | 2750 | 797 |
| ***10*** | XI1 | 55 | 2428 | ***39*** | GND | 2750 | 897 |
| ***11*** | P2.7 | 55 | 2326 | ***40*** | GND | 2750 | 997 |
| ***12*** | P2.6 | 55 | 2224 | ***41*** | P0.3 | 2750 | 1097 |
| ***13*** | P2.5 | 55 | 2122 | ***42*** | P0.2 | 2750 | 1197 |
| ***14*** | P2.4 | 55 | 2020 | ***43*** | P0.1 | 2750 | 1297 |
| ***15*** | VDD | 55 | 1918 | ***44*** | P0.0 | 2750 | 1397 |
| ***16*** | VDD | 55 | 1817 | ***45*** | PWM0 | 2750 | 1830 |
| ***17*** | P2.3 | 55 | 1715 | ***46*** | VDD | 2750 | 2265 |
| ***18*** | P2.2 | 55 | 1613 | ***47*** | VDD | 2750 | 2365 |
| ***19*** | P2.1 | 55 | 1511 | ***48*** | GND | 2750 | 2465 |
| ***20*** | P2.0 | 55 | 1409 | ***49*** | GND | 2750 | 2565 |
| ***21*** | GND | 55 | 1307 | ***50*** | PWM1 | 2750 | 2997 |
| ***22*** | GND | 55 | 1206 | | | | |
| ***23*** | AVDD | 55 | 811 | | | | |
| ***24*** | DACOUT | 55 | 709 | | | | |
| ***25*** | MICIN2 | 55 | 607 | | | | |
| ***26*** | AMPCOM | 55 | 505 | | | | |
| ***27*** | MICIN1 | 55 | 403 | | | | |
| ***28*** | VCM | 55 | 302 | | | | |
| ***29*** | VREF | 55 | 200 | | | | |

Notes:

1. Coordinates are in microns (um), rounded to nearest um.
2. Coordinates are of the center of the bonding pad opening (70um).
3. Coordinate (0,0) is the lower left corner of the die.
4. Die size with scribe and seal ring is 2805 um x 3415 um.
5. **No external die substrate tie is required. However, a substrate tie to ground is preferred.**

## Mechanical Data

***LQFP 100 PLASTICQUAD FLATPACK (14x14x1.4 mm)***



| Symbol | Dimension in mm | | | Dimension in inch | | |
|---|---|---|---|---|---|---|
| | Min | Nom | Max | Min | Nom | Max |
| **A** | - | - | 1.60 | - | - | 0.063 |
| **A1** | 0.05 | - | 0.15 | 0.002 | - | 0.006 |
| **A2** | 1.35 | 1.40 | 1.45 | 0.053 | 0.055 | 0.057 |
| **b** | 0.17 | 0.22 | 0.27 | 0.007 | 0.009 | 0.011 |
| **b1** | 0.17 | 0.20 | 0.23 | 0.007 | 0.008 | 0.009 |
| **c** | 0.09 | - | 0.20 | 0.004 | - | 0.008 |
| **c1** | 0.09 | - | 0.16 | 0.004 | - | 0.006 |
| **D** | 15.85 | 16.00 | 16.15 | 0.624 | 0.630 | 0.636 |
| **D1** | 13.90 | 14.00 | 14.10 | 0.547 | 0.551 | 0.555 |
| **E** | 15.85 | 16.00 | 16.15 | 0.624 | 0.630 | 0.636 |
| **E1** | 13.90 | 14.00 | 14.10 | 0.547 | 0.551 | 0.555 |
| e | 0.50 BSC | | | 0.20 BSC | | |
| **L** | 0.45 | 0.60 | 0.75 | 0.018 | 0.024 | 0.030 |
| **L1** | 1.00 REF | | | 0.039 BSC | | |
| **R1** | 0.08 | - | - | 0.003 | - | - |
| **R2** | 0.08 | - | 0.20 | 0.003 | - | 0.008 |
| **S** | 0.20 | - | - | 0.008 | - | - |
| □ | 0º | 3.5º | 7º | 0º | 3.5º | 7º |
| **□1** | 0º | - | - | 0º | - | - |
| **□2** | 12º TYP | | | 12º TYP | | |
| **□3** | 12º TYP | | | 12º TYP | | |

Notes:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Falls within JEDEC MS-026 BBC

## Ordering Information

| Part | Shipping P/N | Description |
|---|---|---|
| RSC-464 Die | (ROM specific) | Tested, Singulated RSC-464 die in waffle pack |
| | | |
| RSC-464 100LQFP | (ROM specific) | RSC-464 100 pin 14 x 14 x 1.4 mm LQFP |

# The Interactive Speech™ Product Line

Sensory's **Interactive Speech™** product line makes consumer electronics more intelligent by enabling them to talk and hear with speech synthesis, voice recognition, and other advanced audio and interactive technologies. It is designed for integration into cost-sensitive consumer electronic applications such as home electronics, smart toys, music players and personal communication devices. The hardware line includes the award-winning RSC-4x family of mixed signal processors, the *VR Stamp*™ 40-pin DIP module, and the SC-691 music and speech synthesis slave processor. Embedded software options include our FluentSoft™ Recognizer, which enables speech recognition on non-Sensory processors and DSPs. Sensory's BlueGenie™ Voice User Interface, the first Voice Recognition and Synthesis option for BlueTooth enabled devices, offers user friendly control of headsets, music players and other BT devices requiring hands-free operation.

### *RSC Microcontrollers and Tools*

The RSC product family contains low-cost 8-bit speech-optimized microcontrollers that are fully integrated and include A/D, pre-amplifier, D/A, RAM, and ROM circuitry. With Sensory's FluentChip™ firmware, the RSC family offers speech recognition, speaker verification, speech and music synthesis, voice recording and playback, and an entire suite of interactive robotic and sonic networking technologies. The family is supported by a complete suite of evaluation and development toolkits that include the ability to quickly create speaker independent recognition sets in many languages.

### *Speech Recognition Modules and Tools*

The VR Stamp™ is a complete speech recognition module based on the RSC-4x and is ideal for fast design and easy production. A low-noise audio channel and standardized 40-pin DIP footprint allow rapid prototyping, less debugging, and shorter time to market. The *VR Stamp Toolkit* includes everything needed to get started today, including VR Stamps, Module Programming Board, sample applications, and a complete set of development tools featuring the Phyton IDE and limited-life C compiler, QuickSynthesis™ 4 and Quick T2SI-Lite™ speech tools.

### *SC6 Slave Processor and Tools*

The SC-691 is a standard slave synthesizer that accepts compressed speech data from other microprocessors or microcontrollers and converts it to speech. The chip operates up to 12.32 MIPS, and provides high-quality, low data-rate speech compression and MIDI music synthesis, with unlimited speech duration using external memory. Sensory offers hardware and software tools for analyzing speech files, editing speech data and generating coded speech.

### *FluentSoft™ Recognizer*

The FluentSoft™ Recognizer is the engine powering the FluentSoft™ SDK. It provides a noise-robust, large-vocabulary, speaker-independent solution with continuous digit recognition and word-spotting capabilities. This small-footprint software recognizes thousands of words and runs on non-Sensory processors including Intel XScale, TI OMAP, and ARM9, and supports operating systems such as MS Windows, Linux, and Symbian.

### *BlueGenie™ Voice User Interface*

The BlueGenie Voice Interface software suite runs on CSR's BC-5 MM Kalimba DSP, and enables manufacturers of Bluetooth products to integrate full voice control and synthetic speech output without the need for visual displays or complex user interfacing. It frees designers to pack functionality onto small form factor Bluetooth devices and answers consumer demand for a truly hands-free experience.



**575 N. Pastoria Ave, Sunnyvale, CA 94085**
Tel: **(408) 625-3300** Fax: **(408) 625-3350**