

## Getting Started with the RSC-464

This guide is a design note that describes the steps Sensory recommends for building an application for the RSC-464. It points out the major differences in application development between that chip and others in the RSC-4x family. It assumes familiarity with Sensory and Phyton development tools and experience using the RSC-4128.

### RSC-464 Characteristics

The major characteristics of the RSC-464 that influence development are:

- ROM is on-chip only, limited to 64K bytes total (custom ROM mask)
- No external memory bus (external data must use serial memory)
- Internal RAM is smaller than the RSC-4128
- I/O Port 1 is not available

The advantages of the RSC-464 are lower cost and simpler designs; these advantages are accompanied by some restrictions in capability. If your application requires ANY of the following capabilities then it would NOT be a candidate for the RSC-464:

- More than 1 onchip SDSV template
- More than 2 SDWS templates
- Large T2SI sets (more than approximately 10 command words)
- Sensory's Music Player technology
- Parallel external data space
- More than 16 GPIO pins
- Small order quantities (contact Sensory Sales for minimum order quantity)
- More than 64K bytes of code/const space (although see items 10 and 11 below for ways to reduce the size of const space)

### Development Methodology

The RSC-464 is available only in larger quantities with a custom masked ROM, so the RSC-4128 with external code memory is used to emulate the RSC-464. Sensory has taken a number of steps to assure that applications developed in this way will successfully execute when masked in the RSC-464:

- Separate libraries, FC464.MCL and FC4128.MCL
- Separate configuration files, CONFIG.MCA and CONFIG464.MCA
- Embedded linking commands that force all CODE and CONST to be located in the first 64K of ROM.
- Embedded "requires" definitions that assure the correct library is linked
- Warnings issued during development when conditions exist that are not compatible with execution on the RSC-464.
- ROM-embedded "Library ID" byte indicating that the program was linked with FC464.MCL. This value is confirmed at the mask check step.

## Development Platforms

Sensory provides several different development platforms, some of which cannot directly execute RSC-464 code because they cannot run at zero wait states (0-WS). Development of some RSC-464 applications requires a platform that can directly execute RSC-464 code running at 0-WS.

Specifically, the RSC-4x Demo/Evaluation Board, P/N 60-0208, can be used ONLY if:

1. It's combined with a ROM emulator with an access speed of no more than 40 nsec, OR
2. The application under development does not make use of any Sensory technologies that require 0-WS.

If this is your platform and you do not have a fast ROM emulator, contact Sensory for an upgrade.

The following CAN be used to run any RSC-464 application, including with 0-WS:

1. RSC-4x Demo/Evaluation Board, P/N 60-0239
2. RSC-4x Demo/Evaluation Board, P/N 60-0252
3. RSC-4x Target Board, P/N 60-0222, in combination with a Phytion PICE-SE emulator pod
4. RSC-4x Target Board, P/N 60-0242, in combination with a Phytion PICE-SE emulator pod
5. RSC-4x Target Board, P/N 60-0254, in combination with a Phytion PICE-SE emulator pod
6. Sensory VR Stamp
7. RSC-4x Rapid Prototyping Module (RPM)

## Development Steps: Overview and Details

The general approach is to develop the application for the RSC-4128, then “port” it to the RSC-464. This allows tuning the code/const size to get the most out of the 64K bytes available. Sensory recommends following these steps for successful RSC-464 development.

1. Identify one or more sample programs that can be used as a starting point. For example, an application using Speaker Independent recognition of multiple vocabularies might start with the T2SIMATH sample, which illustrates how to combine several T2SI grammars with a single Acoustic Model. Select the sample from \csamples if you will be developing using the C language; otherwise select the sample from \asamples.
2. Build and experiment with the 4128 version of the sample program. This can be done on any platform. It helps familiarize you with the behavior of the program and technology, and gives you ideas of how to implement your application.
3. Copy the entire sample folder and its subfolders to your working directory. You may want to rename files appropriately for your use. Note that all the samples use relative paths, so if your working directory is at the same level as the sample within the Sensory library, the paths will be correct. (for example, c:\Sensory\FC3\_0\_0\projects\myProject).
4. Using the Phytion PICE-SE In-Circuit Emulator or PDS-SE Simulator development platforms, open the appropriate .IDE file for the RSC-4128 configuration you want. For example, the SDF sample can be built with templates stored either onchip or in serial EEPROM, and both versions can be built for either the RSC-4128 or the RSC-464. Note that RSC-464 sample IDEs contain “464” in their name. You may need to revise the library and include paths to point to the correct Sensory folders, and you will have to add any source files names that you changed. You may also need to remove the FC4128.MCL library (with a relative path) and add it back with a correct path.
5. Rebuild the sample in your working directory to confirm you have everything hooked up correctly.
6. Modify the sample as required for your application.

7. Develop any special data files needed: use Quick T2SI for T2SI data files, and Quick Synthesis 4 for SX data files. In Quick T2SI, select RSC-4128 as the target. If you use SXL compression in Quick Synthesis 4, see item 10 below.
8. Refine and debug your code until it is working as intended in the RSC-4128 environment. To this point, these are all normal RSC-4128 development steps. You are now ready to port to the RSC-464 environment.
9. Study the .MAP file to determine the code/const size. Look for lines starting as: "Address area: CODE" and "Address area: CONST". The last entry in CODE that is not "GAP" shows the size. For example, this application uses 189A2h bytes, or about 100K:
 

```
*** OVERLAP ***           0      00018966      000189A2      0000003D      61
*** GAP ***              -      000189A3      0001FFFF      0000765D      30301
```
10. Modify the program to get the total code/const size below 64K. The usual approach is to remove/reduce synthesized speech or to compress it more highly. Note that SXL will produce best speech compression but it requires 0-WS, so using SXL requires a 0-WS hardware platform (see "Development Platforms" above). In this case you must also link with the 0-WS version of FC4128.MCL, called FC4128ZWS.MCL.
11. Another way to reduce const space is to reduce the size of the T2SI sets. In Quick T2SI, on the Vocabulary tab, remove or shorten command phrases. On the Settings tab, make sure that the Matching Criteria for Trigger and Command Phrase Settings are "Match Target phrase exactly". Also on the Setting tab, under Size Settings, choose a smaller Acoustic Model if one is available in your language pack.
12. To execute in your target hardware you will need to install a Phyton PICE on your target board. This can execute either RSC-4128 or RSC-464 code.
13. Remove CONFIG.MCA from your project and replace it with CONFIG464.MCA.
14. Remove FC4128.MCL (or FC4128ZWS.MCL) from your project and replace it with FC464.MCL.
15. Rerun Quick T2SI for your vocabularies, selecting RSC-464 as the target platform. Link these new T2SI objects into your application.
16. Rebuild the RSC-464 application. If the code/const size is still above 64K, you may temporarily defeat the 64K limit by editing ENFORCE\_64K\_LIMIT in CONFIG464.MCA. This will allow building a temporary, too-big executable that will run under the RSC-464 environment, providing another opportunity to reduce the size. The final version, of course, must fit into 64K.
17. The default operation of the FC464.MCL library is to run at 1-wait state, although the 464 SX technologies switch to 0-WS internally. If the application needs even more cycles, edit the WAIT\_STATES line in CONFIG464.MCA to set this constant to 0. The application will switch to 0 wait states during startup and run that way all the time. Running at zero wait-states consumes slightly higher current, but additional cycles are available to the application if needed. NOTE: The developer should take 0-WS operation into account in interrupt service routines, callout and memory handlers and timing loops.

## The Interactive Speech™ Product Line

The Interactive Speech line of ICs and software was developed to “bring life to products” through advanced speech recognition and audio technologies. It is designed for cost-sensitive consumer-electronic applications such as home electronics, home automation, toys, and personal communication. The line includes the award-winning RSC-4x family of mixed signal processors and tools, the *VR Stamp™* 40-pin DIP module and tools, and the SC series of speech and music synthesis microcontrollers. It also includes our suite of software development kits, which are designed to run on non-Sensory processors and DSPs and support most popular operating systems.

### **RSC Microcontrollers and Tools**

The RSC product family contains low-cost 8-bit speech-optimized microcontrollers designed for use in consumer electronics. All members of the RSC family are fully integrated and include A/D, pre-amplifier, D/A, ROM, and RAM circuitry. The RSC family can perform a full range of speech/audio functions including speech recognition, speaker verification, speech and music synthesis, and voice recording/playback. The family is supported by a complete suite of evaluation and development toolkits.

### **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.

### **SC Microcontrollers and Tools**

The SC-6x product family features the highest quality speech synthesis ICs at the lowest data rate in the industry. The line includes a 12.32 MIPS processor for high-quality, low data-rate speech compression and MIDI music synthesis, with plenty of power left over for other processing and control functions. Members of the SC-6x line can store as much as 37 minutes of speech on-chip and include as many as 64 I/O pins for external interfacing. Integrating this broad range of features into a single chip enables developers to create products with high quality, long duration speech at very competitive price points.

### **FluentSoft™ Technology**

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 up to 5,000 words; runs on non-Sensory processors including Intel XScale, TI OMAP, and ARM9 platforms; and supports operating systems such as MS Windows, Linux, and Symbian.

### **3Dmsg™ Technology**

3Dmsg's ([www.3Dmsg.com](http://www.3Dmsg.com)) Animated Speech technology offers animated avatars with advanced speech recognition and synthesis capabilities for use in smartphones, language trainers, and kiosk applications. Facial expressions can be configured to show emotions and lip synchronization can be automatically driven from voice or text data.

### **Important notices:**

Sensory Incorporated (Sensory, Inc.) reserves the right to make changes, without notice, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Sensory, Inc. assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified. Applications that are described herein for any of these products are for illustrative purposes only. Sensory, Inc. makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

### **Safety Policy:**

Sensory, Inc. products are not designed for use in any systems where malfunction of a Sensory, Inc. product can reasonably be expected to result in a personal injury, including but not limited to life support appliances and devices. Sensory, Inc. customers using or selling Sensory Incorporated products for use in such applications do so at their own risk and agree to fully indemnify Sensory, Inc. for any damages resulting from such improper use or sale.



S E N S O R Y®

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

© 2007 SENSORY, INC. ALL RIGHTS RESERVED.  
Sensory is registered by the U.S. Patent and Trademark Office.  
All other trademarks or registered trademarks are the property of  
their respective owners.

[www.sensoryinc.com](http://www.sensoryinc.com)