White Paper – RX100 Microcontroller Family

Implementing Ultra-Low-Power Design Techniques in RX100 MCU-based Applications

Authors: Carmelo Sansone, Warren Miller (Wavefront Marketing)

Renesas Electronics America Inc.

June 2013

Abstract

The widespread initiative among electronic manufacturers to boost the power efficiency of their products has spurred the development of new microcontrollers. Problem-solving system-design solutions like Renesas' RX100 MCUs save power while delivering strong performance, functions and capabilities. These innovative chips wake up quickly from Sleep mode, consume less current when running, and achieve outstanding overall performance advantages that system engineers can use to create products offering sales-enhancing benefits in world markets.

This white paper highlights the low-power capabilities of 32-bit RX100 MCUs and shows how system engineers can apply them to design battery-powered products within extremely tight power dissipation limits. Topics covered include the RX100 architecture's power-efficient Run mode; its power-controlled High-Speed, Medium-Speed and Low-Speed modes; and its low-power Sleep, Deep Sleep and Software Standby modes. Noteworthy features of on-chip peripheral functions are also discussed.

To provide design perspective for minimizing average power consumption, this paper presents two example applications in which it is essential to provide the longest possible battery lifetime: a flow meter and a remote control device. Each takes advantage of a different set of RX100 features and capabilities. The power-saving design methods and options described are useful in a wide range of battery-powered and eco-friendly applications.

Introduction

Demand continues to rise dramatically for electronic products that do more and perform better while consuming less power. Manufacturers serving consumer, home, industrial, office, and medical markets, among others, are striving to produce new battery-based, very low power designs with wireless connectivity that are smaller and more portable and provide new capabilities and features – yet also require less-frequent battery recharges and replacements.

The key enablers of such progress are a new generation of microcontrollers (MCUs) that are much more power-efficient, devices like those in the Renesas 32-bit RX100 MCU family. These chips are designed and fabricated specifically to meet the low-power mandates of the vast range of power-constrained applications that exist today and are anticipated in the future.

RX100 Low-power Features

Renesas RX100 chips are the industry's first 32-bit MCUs to combine a breakthrough power-control technology – our True Low Power™ capability – with exceptional features such as fast wake-up times, zero-wait-state flash, multiple safety functions, integrated USB 2.0 host/device support and OTG support. These new MCUs extend the True Low Power advantages previously introduced in our RL78 MCUs, moving them up to the 32-bit architecture level to give system engineers a broader range of device scalability.

RX100 MCUs deliver optimized combinations of ultra-low-power consumption, on-chip connectivity and superior performance at price points that are ideal for high-volume embedded systems. Especially, these microcontrollers are the best choices for low-end 32-bit applications like mobile healthcare devices, smart meters, and security systems, as well as sensors, detectors and other elements of industrial-control systems and building-automation equipment.

Major low-power features and characteristics of Renesas RX100 MCUs include the following:

- Exceptional RUN-mode power efficiency: 100µA/MHz
- Ultra-fast wake-up time: 4.8µs
- Superior architecture: 3.08 Coremarks/MHz performance
- Six operating modes, plus numerous other design options for saving power
- Standard and advanced on-chip peripherals: ADC, LVD, RTC, USB, and more.

32-bit Architecture with 'True Low Power' Technology

The upgraded True Low Power capability in RX100 MCUs covers all on-chip peripherals and the Flash memory, allowing the smallest possible power consumption over the devices' entire temperature and voltage range. System design flexibility is maximized across diverse applications via multiple active and power-down modes.

All aspects of MCU design and fabrication are addressed in Renesas' power-saving system solutions, giving customers applying RX100 chips big advantages for meeting power-budget challenges. For example, these 32-bit devices are built with the 130nm low-leakage process technology Renesas has used so successfully in RL78 MCUs.

Multiple power-controlled Run modes (High-Speed, Middle-Speed, and Low-Speed) minimize power consumption when different CPU speeds are needed for various application tasks.

Additionally, three Low-Power modes (Sleep, Deep Sleep and Software Standby), in combination with the short wake-up times from these modes, let system engineers fine-tune the power supply current to specific application requirements.

Other power-saving features are also noteworthy. The Zero-Wait-State Flash memory technology built into RX100 MCUs decreases power consumption because the CPU doesn't have to remain idle while waiting for data fetched from nonvolatile storage. Also, every on-chip peripheral module can be powered off individually, so that those not being used don't waste power.

The advanced clock system incorporated into the RX100 architecture allows the speed of the clocks driving the peripherals to be reduced while the CPU operates at up to its maximum frequency. Moreover, there is a choice of oscillators (HOCO or LOCO) for waking up the CPU, and extra power reductions can be obtained in some situations by utilizing those clocks to replace the Phase-Lock Loop (PLL) main clock.

Importantly, the CPU design and RX architecture are extremely compute-efficient, achieving the highest possible number of instructions per mW. Interrupt latency is only 5 cycles and processing performance is rated at 1.54 DMIPS/MHz and 3.08 Coremarks/MHz.

The architecture's large number of parallel busses makes possible simultaneous movements of data between the CPU core, Flash, SRAM and peripherals. This design feature ensures that no bottlenecks are present when the CPU wakes up from a power-down mode.

It's significant that Renesas employs a truly holistic approach to producing power-saving MCUs (see Figure 1). We are the only microcontroller supplier to do so. By maintaining direct control of all the elements of MCU development and manufacture, our semiconductor technology experts enable the production of high-quality, optimized system-design solutions that customers can apply to implement ultra-low-power products and systems.

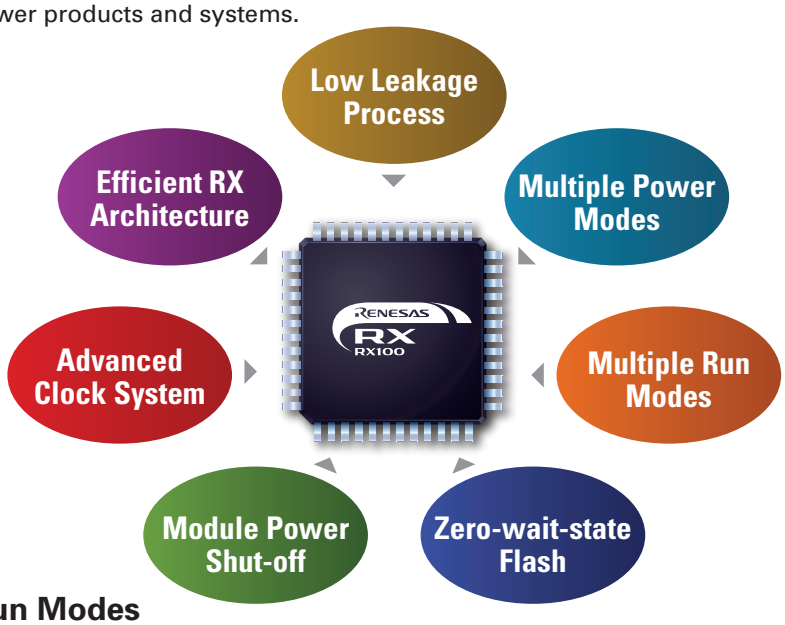

Figure 1: Renesas' holistic approach to obtaining low-power operation and fast wake-up times

Power-controlled Run Modes

With an RX100 MCU, system engineers can tailor the available processing capability and the chip's power consumption to match the computational requirements of diverse application tasks. As previously mentioned, the CPU has three power-controlled Run modes: High-Speed, Middle-Speed and Low-Speed.

Each of these modes makes available a different set of on-chip peripheral modules. Some restrictions apply, though. The availability of some oscillators, the PLL, Flash memory programming and certain peripheral clock frequencies depends on the Run mode selected.

By contrast, the MCU's supply voltage requirements aren't affected by the power-controller Run modes. Operation is always allowed over the device's full 1.8V to 3.6V range. However, the clock frequencies usable in the High-, Middle-, and Low-Speed modes do depend on the supply voltage (see Table 1).

Table 1: RX100's power-controlled Run modes

Power-controlled Operating Mode	Operating Voltage Range	Operating Frequency Range			
		ICLK	FCLK	PCLKD	PCLKB
High-Speed	3.6 to 2.7V	Up to 32MHz	Up to 32MHz	Up to 32MHz	Up to 32MHz
	2.7 to 2.4V	Up to 16MHz	Up to 16MHz	Up to 16MHz	Up to 16MHz
	2.4 to 1.8V	Up to 8MHz	Up to 8MHz	Up to 8MHz	Up to 8MHz
Middle-Speed	3.6 to 1.8V	Up to 8MHz	Up to 8MHz	Up to 8MHz	Up to 8MHz
Low-Speed	3.6 to 1.8V	Up to 32.768kHz	Up to 32.768kHz	Up to 32.768kHz	Up to 32.768kHz

Table 2 lists the clock sources that can be used in each of the three Run modes.

Table 2: Clock sources usable in the power-controlled Run modes

Mode	PLL	HOCO	LOCO	Main Osc.	Sub Clock
High-Speed	Usable*	Usable	Usable	Usable	Usable
Middle-Speed	Usable*	Usable	Usable	Usable	Usable
Low-Speed	Not Usable	Not Usable	Not Usable	Not Usable	Usable

** VCC ≥ 2.4V*

Low-power Operating Modes

Besides the three Power-Controlled operating modes, RX100 MCUs also offer the previously mentioned Low-Power operating modes: Sleep, Deep Sleep and Software Standby. In each of them, different MCU functions are stopped and/or powered down, saving various amounts of current. Here are the details:

Sleep mode: The CPU is stopped with data retained. This reduces the CPU's dynamic current consumption, which is a significant contributor to the MCU's overall operating current. The CPU wakes up from Sleep mode into the Run mode in only 0.21µs at 32MHz.

Deep Sleep mode: The CPU, RAM and Flash memory are stopped, with data retained. At 32MHz with multiple peripherals active, the typical operating current is only 4.6mA. It takes just 2.24µs for the CPU to wake up from Deep Sleep mode and enter Run mode.

Software Standby mode: The PLL and all the oscillators except the sub-clock and IWDT are stopped. Almost all of the RX100's modules – CPU, SRAM, Flash, DTC and peripheral blocks – are stopped, with data retained. The Power-on-Reset (POR) circuit remains operational, though, and if necessary, the IWDT, RTC, and LVD modules can be operated. Current consumption in this mode is from 350nA to 790nA, depending on whether or not the LVD and RTC functions are used. When waking up in the 4MHz Run mode, CPU operation begins after a 4.8µs delay. When waking up in the fast 32MHz Run mode, the wait time extends to 40µs.

Table 3 shows the power consumption levels and wake-up times for the RX111 MCU.

Table 3: RX111 power consumption levels and wake-up times

MCU Configuration															Wake-up Time	Current Consumption (25C, 3.3V)		
Power Mode	CPU Clock	Peripheral Clocks	Mode	Regulator	LVD	HS OCO	HS Ext Osc	PLL	LS OCO	32KHz Ext Osc (RTC)	RAM State	I/O Pin State	Code Source at Wake-up	Wake-up Sources		Frequency	Min (mA)	Typ (mA)
Active (Run)	ON	ON/OFF	High	ON (NVHC)	ON	Clock ON (32MHz)	OFF	OFF	Clock OFF	ON	Active	Active	Flash	Any Interrupt, LVD, POR, Ext Reset	–	32MHz	3.2	10.6
			High	ON (NVHC)												8MHz	1.7	3.7
			Middle	ON (LVHC)												8MHz	1.32	3.5
																4MHz	–	2.15
																1MHz	0.74	1.2
			Low	ON (LVLC)												32KHz	0.00396	–
Sleep	OFF	ON/OFF	High	ON (NVHC)	ON	Clock ON (32MHz)	OFF	OFF	Clock OFF	ON	Active	Active	Flash	Any Interrupt, LVD, POR, Ext Reset	0.21µs	32MHz	1.8	6.4
			Middle	ON (LVHC)					Clock OFF						0.875µs	8MHz	0.9	2.2
			Low	ON (LVHC)					Clock OFF						7µs	1MHz	0.7	1
Deep Sleep	OFF	ON/OFF	High	ON (NVHC)	ON	Clock ON (32MHz)	OFF	OFF	Clock OFF	ON	Active	Active	Flash	Any Interrupt, LVD, POR, Ext Reset	2.24µs	32MHz	1.2	4.6
			Middle	ON (LVHC)					Clock OFF						3.55µs	8MHz	0.7	1.8
			Low	ON (LVHC)					Clock OFF						15.80µs	1MHz	0.6	0.9
Software Standby	OFF	OFF	Standby	ON (LVLC)	ON	Power OFF Clock OFF	OFF	Power ON Clock OFF	Power On Clock OFF	ON	Retain	Retain	Flash (Powered On)	Any External Interrupt Pin, POR, RTC Alarm, Wake-up, Ext Reset	4.80µs (4MHz) 40µs (32MHz)	790nA		
										OFF						450nA		
					OFF					ON						690nA		
										OFF						350nA		

Min: Peripheral clocks all stop, CPU NOOP-Loop, Flash access 25%, Peripheral modules all stop

Typ: Peripheral clocks all running no divider, CPU all command operation, Peripherals modules on – DTC/RSPI 1 channel, MTU 1 channel, CMT 1 channel

Additional RX100 Power-saving Capabilities

Although the Sleep, Deep Sleep and Software Standby modes of RX100 MCUs are very helpful for decreasing the current the chips consume, system engineers can use other techniques to achieve further power reductions. For instance, they can set various clock-signal frequency-division ratios individually. This capability applies to the system clock, peripheral module clock, S12AD clock and Flash clock. It's a valuable design option when application requirements differ between function blocks.

Also, each peripheral module has a separate Stop control bit. This feature allows software to exercise individual control of the MCU's on-chip functions to further reduce dynamic current.

Application Examples

The remainder of this white paper presents two typical example applications that utilize specific low-power features of RX100 MCUs: a flow meter and a remote control device. Both of these designs are similar in that they employ serial interfaces. However, they differ fundamentally in the MCU characteristics needed to fulfill their application's operational requirements, making them useful for exploring and explaining various ultra-low-power system-design techniques.

Each application discussion tells what the system does and describes its low-power/battery-lifetime requirements. Then it highlights appropriate low-power design techniques and/or options, and explains how to best apply key features of RX100 MCUs. Performance data are used to calculate the example design's average current usage and show the resulting battery-lifetime.

To reduce duplication, MCU features and design techniques described in the remote control example build upon technical issues discussed in the flow meter section. So maximum insight is gained by reading both examples in sequence.

Battery-lifetime computations focus on the current contributed by the RX100 MCU. For clarity and brevity, the current drain of external components isn't considered in the examples. Various design techniques are helpful for applications in which external devices must be considered; however, they are beyond the scope of this white paper.

One additional system-design issue must be mentioned here: All charged batteries eventually lose their charge through their internal resistance, even without an external load. The application discussions in this white paper don't take into account the self-discharge phenomenon. System engineers should incorporate relevant data supplied by the battery's manufacturer into comprehensive battery-lifetime calculations.

Flow Meters

Modern flow meters are evolving from simple, manually read mechanical units to microcontroller-based electronic versions with wireless connectivity. Fancier designs offer flexible monitoring and data communications, allowing control by the utility company's central control system. Such advanced features must be implemented despite the fact that the meter is always on. Therefore, the electronics must consume only miniscule amounts of power on average. Battery operation is the norm, since AC power seldom is available to run the meter's MCU-based circuits. Typical design specifications mandate a battery lifetime of over 20 years.

The primary low-power requirements of an electronic flow meter can be grouped according to the major functions the unit performs. Most of the time, the meter's MCU operates in a low-power mode, with just the Real-Time Clock (RTC) and Low Voltage Detector (LVD) running. Also, it's generally recommended that the SRAM be kept active so that it can store intermediate processing results, eliminating the need to continually write data to the Flash memory.

Periodically the meter wakes up and makes a flow measurement. Key metrics (used for billing) are saved in non-volatile memory so they won't be lost if power is interrupted. Communications with the central control system are implemented via the serial transceiver whenever data has to be collected or updates implemented. Additionally, the voltage levels of the battery are checked regularly to manage the MCU's operating modes.

Example Flow Meter Design

The following discussion of a typical flow meter design looks in detail at key aspects of its operation to obtain information essential for estimating battery lifetime. For this example, the data are derived from an implementation based on an RX111 chip (see Figure 2).

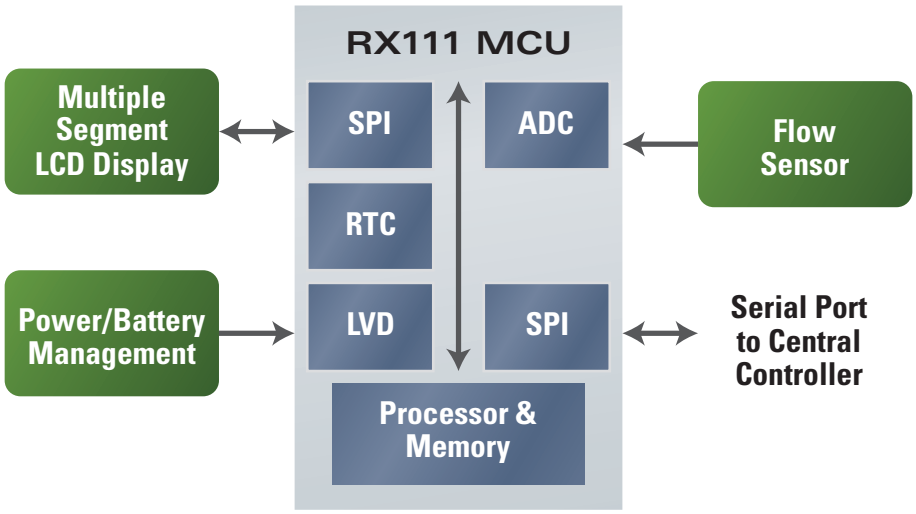

Figure 2: Flow meter example application

The RX111 processor, memory and peripherals integrate most of the flow meter's functions. The main peripherals used are the Analog-to-Digital Converter (ADC) that measures the output of the flow sensor, an SPI port that connects to the central controller that collects data from a variety of meters, and another SPI port that drives the LCD that shows flow-usage data and system status. Additionally, the MCU's RTC keeps accurate track of the time each measurement is made, and the Low-Voltage Detector (LVD) continuously monitors the voltage of the meter's battery.

Functional Description

In order to estimate the power use and battery lifetime of the flow meter, it's necessary to identify key facts: the main functions that the meter's MCU has to execute, what modules are utilized in performing the tasks, how often the functions are performed, how long it takes to execute those functions, and the current the MCU uses in handling those tasks. Some of the MCU's on-chip peripherals, like the RTC and LVD, operate continuously, while others, like the ADC or SPI ports, are needed for only short periods of time. Details of the example flow meter's functions are described below:

Battery Monitor: This function checks the voltage level of the battery, producing information that's included in the operational-status data sent periodically to the central control system. The battery monitor can also perform a tamper-detection task to determine whether or not the meter has been subjected to an attack.

The measured battery voltage (and its variation over time) is used to adjust the frequency of operation of various meter functions, because by extending the time between operations as the battery level drops, battery lifetime can be extended.

When battery voltage gets too low, the monitor can initiate a 'going off air' signal that causes vital flow-usage and system data to be stored in Flash memory for later retrieval and diagnostics. This operation is seldom needed, though, so it isn't included in the battery lifetime calculation. (System engineers are urged to set the battery monitor's low-level warning trip-point high enough to ensure that sufficient power remains for the MCU to execute any 'last gasp' data collection, security and safety tasks.)

Flow Monitor: The MCU's ADC reads the output of the sensor to accurately measure the flow rate. The data it provides subsequently must be processed to determine actual billable metrics. The requisite computations are well within the processing capability of the RX111 MCU.

Send Update: This function communicates key data (flow rate, battery level, quality of service, etc.) to the central control system. The time between Send and Receive transmissions can be extended as the battery runs down in order to conserve battery power.

Receive Update: The flow meter receive function is activated on an as-needed basis by the central controller. When the meter receives update data, its MCU has to be able to quickly perform important housekeeping functions before executing the update.

A basic function table, like the one shown below in Table 4, is a convenient way to organize these basic operating functions. It shows the active peripherals associated with each meter function, how often the task is performed, and how long it takes the MCU to execute each function.

Table 4: Characteristics of main functions of the example flow meter

Function	Operational Characteristics		
	Peripherals Active	Avg. Number of Executions per Minute	Processing Time Estimate
Flow Monitor	RTC, LVD, ADC	60 (1s interval)	15µs
Battery Monitor	RTC, LVD	1 (1 min. interval)	30µs
Send Update	RTC, LVD, SPI	0.1 (10 min. interval)	1000µs
Receive Update	RTC, LVD, SPI	0.1 (10 min. interval)	2000µs

The data in Table 4 are estimated values for the MCU processing time required for each function, instead of measured values, because the functions weren't actually implemented. Still, they are 'best conservative guesses' based on similar functions designed in other applications and are valid for this paper's power-use calculations.

Implementation Options and Low-power Design Techniques

Several implementation options typically exist when designing a low-power MCU-based system. One common software option is to place the system in a power-down mode, and then turn it on after a specified time interval has elapsed. This option is called a Periodic Wake-up.

Applying the Periodic Wake-up approach to the example flow meter design, the following operational schedule was specified: Software wakes up the CPU every second. The MCU has to execute the Send Update function at each 10-minute interval, perform the Battery Monitor task at each 1-minute interval and activate the Flow Monitor function at each 1-second interval.

The Receive Update function is different, though. It's an exception to the operational schedule because it executes asynchronously to the MCU's time-base whenever the central control system requests it. For the purposes of this application example, the worst case is that the Receive Update function is performed about once every 10 minutes. Thus, for the meter's current-consumption

calculations, Receive Update is considered to be a regular function with a 10-minute interval. Here are more details about the main tasks the meter's microcontroller performs:

Generating Periodic Wakeups: Because the Real-Time Clock operates continuously in this meter design, it provides a convenient, power-efficient method for generating the 1-second periodic wake-up signal. The 128-Hz clock that drives the RTC is derived from the Sub-clock (XCIN) 32.768-kHz input. Counters in the RTC produce accurate time signals (year, month, week, day, hour, minute and second) for up to 99 years, making automatic leap-year corrections. The MCU's Alarm mode (ALM) can generate an interrupt on the year, month, date, day-of-week, hour, minute or second.

Another interrupt source, the Periodic Interrupt (PRD), is convenient for initiating shorter time periods because it can generate an interrupt every 2 seconds, 1 second, 1/2 second, 1/4 second, 1/8 second, 1/16 second, 1/32 second 1/64 second or 1/256 second. The example flow meter design utilizes the 1-second PRD interrupt for operational timing.

Monitoring Flow: Once every second, the MCU's ADC converts the output of the external Flow Sensor to produce digital flow data. The ADC converter is turned on via software prior to each measurement. This keeps power dissipation low, since the converter adds 0.66mA to the current consumption when the MCU is running at 32MHz. At that clock speed, the ADC has to be enabled for 3µs to make a measurement: 1µs to enable the A/D, 1µs to perform conversion, and a 1µs delay before the converter is subsequently disabled. At 32MHz, the RX111's wake-up time into the Run mode is 40µs. This time must be added to the 15µs it takes the CPU to process the flow data from the ADC.

Measuring Battery Level: The Low Voltage Detector in RX100 MCUs has two separate voltage detection circuits. The LVD1 circuit measures the battery voltage (VCC). It can compare this voltage to ten different voltage 'steps', ranging from 1.86V to 3.1V. By contrast, the LVD2 circuit can compare an external voltage source to four different voltage 'steps', ranging from 1.8V to 2.9V.

In this flow meter design, VCC is checked every minute to monitor its condition using the LVD1 module to get the most accurate measurement. It generates an interrupt if the level begins to approach the RX111's specified lower voltage limit of 2.7V for 32MHz operation. The MCU stores the measured battery voltage and, if necessary, sends an alert to the utility's central control system during the next Send Update operation. The battery measurement function can run at 1MHz, so its associated wake-up time is only 4.8µs. Its processing time (at 1MHz) is estimated to be approximately 35µs. Thus, the total active time for this function is about 40µs.

Sending Updates: Every 10 minutes, the Send Update command uses the SPI peripheral to transmit data to the central control system. To calculate the energy used by the meter design, an engineering assumption is made that it requires 1000µs to process and transfer the data.

Receiving updates: Every 10 minutes, the Receive Update command uses the SPI peripheral to receive data from the central control system. It is assumed that 2000µs is needed to wake, receive and process the data.

Average MCU Current Consumption Calculation

Figure 5 shows the execution time and current drain of each of the flow meter's functions. The 10.6mA current consumption number (High-Speed Run mode; Table 3) is used here because the RX111 MCU's CPU is active and some of the chip's built-in peripheral functions may also be active. When the ADC is active, it adds 0.66mA to the current consumption.

Table 5: Function characteristics for flow meter example design

Function	Battery Lifetime Estimate (RX111, typ.) for Example Flow Meter						
	MCU Mode	Execute Period	Execute Time (µs)	% Cycle Active	Peripherals Active	Current Drain (mA)	Average Current (µA)
Wait	Software Standby	1sec	NA	100%	RTC, LVD	0.00079	0.7900
Flow Monitor	Run, 32MHz	1sec	3	0.000300%	RTC, LVD, ADC	11.3	0.0339
			55	0.005500%	RTC, LVD	10.6	0.5830
Battery Monitor	Run, 1MHz	1min	40	0.00007%	RTC, LVD	1.2	0.0008
Send Update	Run, 32MHz	10min	1000	0.000167%	RTC, SPI, LVD	10.6	0.0177
Receive Update	Run, 32MHz	10min	2000	0.000333%	RTC, SPI, LVD	10.6	0.0353
Total							1.4607

To determine the meter's battery lifetime, the individual average current for each function is calculated by multiplying the Current Drain by the Percent Cycle Active. Results are shown in the right-hand column in Table 5. The sum of these contributions is the total average MCU current (ICC): 1.46µA.

Of the major contributors to the meter's average current consumption, the Software Standby mode current, 0.79µA, accounts for about 54% of the 1.46µA total, while the Flow Monitor function consumes 0.62µA, or approximately 42% of the total.

In applications like this that have relatively long periods of inactivity, the current consumed in the Software Standby and Run modes generally accounts for most of the average MCU current. Thus, it's important that the MCU used in the design has excellent low-power characteristics in both of these modes.

Battery Lifetime Calculation

For this example application, the meter's battery pack is assumed to have a capacity of 300mAh and provides approximately 3V for most of its life. Given that information, the battery lifetime is computed by dividing the average MCU current into the battery capacity, as indicated below:

300,000µAh/1.46µA = 206,243 hours, or 23.5 years.

The calculation reveals that the battery lifetime of the RX111-based flow meter exceeds the specified 20-year requirement. This result clearly demonstrates the system design advantages gained by applying the exceptional low-power characteristics of an RX100 MCU.

Summary

- The advanced low-power characteristics of RX100 MCUs make them excellent solutions for flow meters and similar applications that require battery operation. Among the device features most helpful in such uses are the following:
  - The power-efficient Run mode
  - Very low Software Standby current
  - The fast wake-up time from Software Standby mode
  - Low power dissipation for RTL and LVD peripherals
  - Power-efficient processing at slower clock frequencies

Remote Control Device

Handheld remote control devices for consumer electronics products, garage door openers, industrial lighting systems and automotive applications have to run on batteries for as long as possible. Typically, the goal is to have them operate for four years before the batteries must be replaced. As a result, reducing power consumption is a major system design concern. Also, higher-end remote control devices have touch-sensitive displays with full-featured user interfaces. They apply intelligent power management to achieve long battery life.

A typical microcontroller-based handheld remote control unit uses an RF interface to communicate with the controlled equipment. Its multiple-segment display provides user feedback and conveniently shows the current time. The device's electronic circuits respond to commands entered via a keypad.

Battery-lifetime related issues dominate the implementation of a remote control unit. Major system-design factors are the amount of time the application spends in the active mode and the amount of time the MCU spends in its low-power mode. Those factors, along with the corresponding MCU's current-drain characteristics, determine battery lifetime.

The remote control unit's MCU waits in a low-power mode for interrupts from three sources: the RTC (to update the time on the display), the keypad (to respond to a button being pressed) and a battery voltage monitor (to check the status of its power supply).

After the MCU exits from the low-power mode, it manages the display and keypad activity used for executing user commands. It sets a 'Battery Low' indication in the display after receiving a power-low interrupt. If the battery level drops too low, the MCU shuts down the remote control function to avoid sending incorrect commands to the equipment being controlled.

Example Remote Control Device Design

The remote control design described in this example application uses a 32-bit RX100-series device as the main controller – again, the RX111. That MCU connects to an external multiple-segment display, a keypad and an RF transceiver, per the block diagram shown in Figure 3.

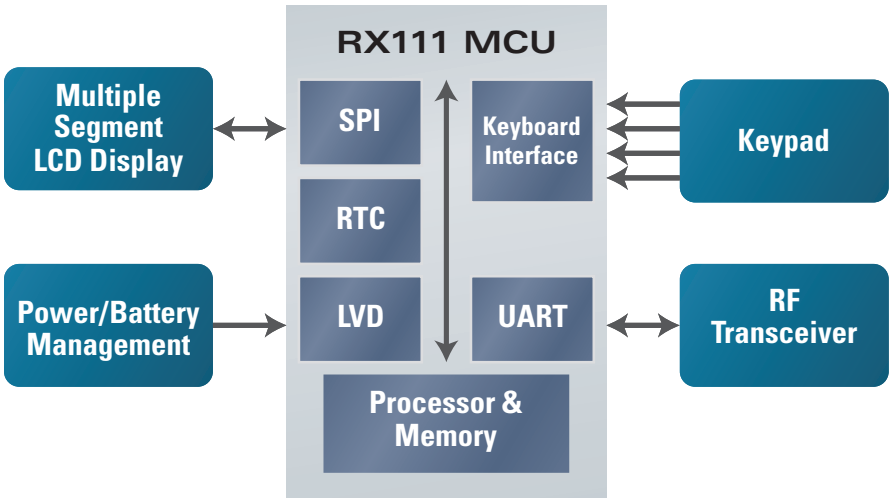

Figure 3: Remote control unit (example application #2)

The RX111's CPU, memory and peripheral modules perform most of the remote control unit's functions. The main on-chip peripherals used are the Keypad Interface that monitors keypad activity, the UART that handles communications with the RF transceiver, and the SPI that sends information to the multiple segment display. Additionally, the Real Time Clock (RTC) keeps accurate track of the time, and the Low Voltage Detector (LVD) monitors the battery voltage.

Most of the time, the MCU remains in its Software Standby mode, in which the CPU and most of the clocks and peripherals are disabled. The chip idles when waiting for an interrupt to wake the CPU and begin some activity.

Two different software design approaches can be applied in this remote control. They're described in the discussion that follows, and data on the resulting battery lifetimes they produce is presented. In one implementation, the CPU wakes up from Software Standby mode into a low-frequency Run mode state. In the other, the CPU wakes up into a full-speed (32MHz) Run mode.

The discussion below aims to answer the following question: At what point does the higher current drain – but reduced active time – of a faster processing mode result in a lower average MCU current than the lower current drain – but longer active time – of a slower power-saving mode?

Functional Description

In order to estimate the power use and battery lifetime of the remote control design, important timing and power components of the remote control unit's functions must be determined. A detailed understanding of a typical user's activity model and usage requirements has identified and quantified the application parameters explained below:

Time Update: The display is updated 120 times a minute; i.e., once every half-second.

Battery Voltage Monitor: The Low Voltage Detector operates all of the time. It wakes up the CPU when the battery level falls too low, but this operation happens so infrequently that its active current component can be ignored. Nevertheless, the LVD's current drain is included in the Software Standby current drain during the time when the MCU is waiting for a battery-low interrupt.

Keypad Operation: The keypad function manages the keypad and determines which equipment control function the user wants to manipulate and in what way. The design assumption here is that the keypad is pressed at an average rate that doesn't exceed 60 key presses per minute over the course of a full day. Application software checks for stuck keys and other obvious keypad errors, reducing the chance of accidental operation that would unnecessarily drain the batteries.

RF Operation: This function enables RF transmission whenever the remote control unit has to execute a user command. The remote control design assumes that this occurs no more than 60 times a minute on average (based on the keypad operation assumptions).

Table 6 summarizes these functions. It shows the active peripherals associated with each function, along with the number of times the function must be executed per minute. The target design requirements for the processing time for each function are based on previous experience with similar functions. They're needed for power-use calculations.

Table 6: Remote control function characteristics

Function	Operation Characteristics		
	Peripherals Active	Avg. Number of Executions per Minute	Processing Time Design Requirement
Wait (with Battery Voltage Monitor active)	RTC, LVD	NA	NA
Time Update	RTC, LVD	120	40µs
Keypad Operation	RTC, LVD, Keypad	60	60µs
RF Operation	RTC, LVD, UART	60	4000µs

Implementation Options and Low-Power Design Techniques

A common system-design method for saving power in battery-based products is to optimize the CPU's operating frequency. In some cases, it's best to run the CPU at the fastest possible speed in order to reduce the amount of time that active-mode current is required. In other situations, it is best to run the CPU at a slower speed to minimize the active-mode current drain.

To illustrate the design tradeoffs involved in these very different strategies, both of these options are examined in this example design. As mentioned previously, the objective is to determine the situations in which one approach is typically better than the other.

Table 7 shows the RX111 MCU's current drain at various CPU operating frequencies measured at 25°C with a 3.3V supply. The 'CPU Current Drain' column is for simple CPU operations, while the 'CPU and Peripheral' column shows the current drain when operations require a combination of CPU and on-chip peripherals.

Table 7: RX100 current drain vs. CPU operating frequency

CPU Operating Frequency	Minimum Operating Voltage	Current Drain (CPU)	Effective Current Drain (CPU)	Current Drain (CPU and Peripheral)
32MHz	2.7V	3.2mA	3.2mA	10.6mA
8MHz (High-Speed)	2.4V	1.7mA	6.8mA	3.7mA
8MHz (Middle-Speed)	1.8V	1.32mA	5.28mA	3.5mA
1MHz (Middle-Speed)	1.6V	0.74mA	23.7mA	1.2mA
32kHz (Low-Speed)	1.8V	3.96µA	3.96µA	–

One important fact revealed by Table 7's data is that, for purely compute-bound routines, using the higher frequency always results in a lower effective current drain. Faster processing requires more current, but tasks are completed in less time. For example, the current drain at 8MHz (High-Speed) is only 1.7mA, but a computation requires 4x the time when compared to 32MHz. The effective current drain is thus 6.8mA (or 4x 1.7mA). Similar comparisons of Effective Current Drain show the benefit or running the CPU faster for improved power efficiency for compute-bound routines.

In general, with an RX100-type MCU the most power-efficient strategy is to execute compute-intensive routines with the CPU running at 32MHz. Similarly, if performance is limited by battery voltage, it's usually best to run the CPU at the maximum frequency allowed.

Another system-design issue should be considered though: The wake-up times from Software Standby mode must be added to the processing times. When those delays are taken into account, a power-usage analysis may reveal that in some cases it is more efficient to wake up into a lower-frequency Run mode that has a shorter wake time.

Compare, for instance, the 40µs it takes to wake up into a 32MHz Run mode to the 4.8µs needed to wake up into a 4MHz Run mode. Be aware that during that 35.2µs difference in wake-up times, the CPU could be executing application tasks.

For applications that don't spend much time doing computations – like those that depend on the timing of external signals – the CPU is likely to spend a lot of time in a low-power Run mode with little to do. For example, the UART might be running at rates slow enough that the CPU is usually just waiting for serial data to be received. In such cases, excess power is consumed unless a significant amount of processing can be overlapped during the time spent waiting for data reception. Typical processing tasks that might be handled include error checking and other communications overhead chores. The main functions performed by the MCU in the example remote control device are described below:

Making RF transmissions: The UART connected to the RF transmitter is assumed to operate at a 128Kbit/sec rate using an 8-bit word, so a complete command transmission takes 12,000µs.

A word has to be presented to the UART at greater than a 1MHz rate to satisfy the transmitter's bandwidth. Additionally, there must be sufficient CPU computing capability to process the message data during transmission.

This design assumes that the CPU is running during the entire transmission time in order to process data. Significantly, the RX111's CPU could operate at a clock rate as slow as 8MHz and still provide sufficient processing bandwidth. Therefore, it is clocked at that speed in the Middle-Speed power-controlled Run mode and consumes only 1.32mA, as shown in the CPU column of Table 7. This design choice is appropriate because only simple CPU operations are being performed and a minimum number of peripheral blocks are enabled. The 1.32mA current consumption at 8MHz is about 60% lower than the 3.2mA that the CPU would consume if it were running at 32MHz.

Additionally, as Table 7 indicates, at 8MHz the RX111 MCU can operate down to 1.8V. This allows a longer battery lifetime compared to 32MHz operation because the 32MHz speed requires a minimum operating voltage of 2.7V. The current consumed by the example remote control when the CPU runs at 8MHz and 32MHz are calculated later in this section.

Updating the Time display: Every half-second, the MCU updates the display with the time, operating mode status and other user-interface data. The RTC generates an interrupt twice every second and wakes-up the CPU if it's in the Software Standby low-power mode. As previously pointed out, the RX111 transitions from that mode into the 32MHz Run mode in 40µs. At that speed, the CPU needs 60µs processing time for updating the display. Thus, at 32MHz, the total current drain is 3.2mA for 100µs (40µs wake-up time plus 60µs processing time).

By contrast, if the CPU is operating with a slower clock, the display processing time will be longer, but the current drain will be less. At 8MHz, the RX111's total current drain is 1.32mA for 245µs (4.8µs wake-up time plus 240µs processing time), since processing takes 4 times as long at 8MHz as it does at 32MHz.

Calculations of the average current consumption in both operating conditions reveals that the CPU's 320mAµs consumption for updating the display when running at 32MHz is slightly less than its 323mAµs contribution when operating at 8MHz. (A minor simplification made here is that the current drain during the wake-up period is the same as it is in the Run mode.)

Handling keypad inputs: The design for this example remote control unit assumes that a key press occurs on average 60 times a minute. The keypad routine can be executed in 100µs when the CPU is running at 32MHz. Given that the wake-up time from the Software Standby mode into the 32MHz Run mode is 40µs and the Run mode operating current at that speed is 3.2mA, the total current consumption of the keypad routine at 32MHz is 448mAµs.

However, the wake-up time into the 8MHz Run mode is only 4.8µs. At that CPU speed, the keypad routine executes in 400µs and the operating current is 1.1mA. For the keyboard routine, then, the CPU consumes 445mAµs running at 8MHz, just slightly lower than the 448mAµs it consumes at 32MHz.

Average MCU Current Consumption Calculation

Requirements for the execution time and the current drain for each function of the example remote control unit are shown in Table 8 on the next page. Data for CPU operation at 32MHz and 8MHz allow a comparison of those design options.

Table 8: Function characteristics for remote control example design

Function	Battery Lifetime Estimates (RX111, typ.) for Remote Control Device						
	MCU Mode	# Per 1 min Cycle	Active Time (µs)	% Cycle Active	Peripherals Active	Current Drain (mA)	Average Current Contribution (µA)
Wait	Stop	NA	NA	100%	RTC, LVD	0.00079	0.7900
Time Update	32MHz	120	100	0.02000%	RTC, LVD	3.2	0.6400
	8MHz		245	0.04900%	RTC, LVD	1.32	0.6468
Keypad Update	32MHz	60	140	0.01400%	RTC, LVD	3.2	0.4480
	8MHz		404.9	0.04049%	RTC, LVD	1.32	0.5345
RF Transmission	32MHz	60	12000	1.20%	RTC, UART, LVD	3.2	38.4000
	8MHz					1.32	15.8400
Total	32MHz	1.23% Run cycle time					40.28
	8MHz	1.29% Run cycle time					17.81
Battery Life	32MHz	1200mAh					3.4 years
	8MHz						7.7 years

Battery Lifetime Calculations

Each function in Table 8 shows the computation at both the 32MHz CPU operating frequency and the 8MHz operating frequency. The number of times the function is executed per minute and the estimated active times are combined to create the '% Cycle Active' numbers. Multiplying those data by the current drain yields the contribution by each routine to the average current consumed during the 1-minute cycle.

Subsequently, those contributions are summed to generate the MCU's total average µA/sec. Finally, the total averages are divided into the battery capacity to obtain the battery lifetime. (The 1200mAh capacity listed assumes that the remote control uses two 680mAh batteries, and it includes an allowance for the battery's self-discharge current.)

The computations reveal that when operating at 8MHz, the battery lifetime achieved by the RX111-based example remote control design is about 2.3 times what it is when the CPU is operating at 32MHz. Clearly, slower is better in this case, because the application is not compute-intensive.

Interestingly, the RF Transmission function makes by far the largest contribution to the battery lifetime difference shown in Table 8. Because this function is I/O bound, the execution time depends on the serial transmission of data. If the lower operating frequency can handle the required data rate, that option can provide a big power-efficiency advantage.

This example design shows that by identifying opportunities to save power by running at a reduced CPU frequency, it's possible to extend battery life in many I/O-oriented products.

Summary

- The advanced low-power characteristics of RX100 MCUs make the devices excellent solutions for handheld Remote Control units and similar products that are battery-based because these Renesas chips offer the following features:
  - A power-efficient Run mode
  - An ultra-low Software Standby current
  - Low power consumption at slower Run frequencies
  - Fast wake-up times
  - Low power operation of RTC and LVD peripherals
- The lower-frequency Run modes of RX100 MCUs allow longer battery lifetimes if the primary application routines have a fixed execution time; i.e., one that's not determined solely by CPU performance.

Conclusion

The MCUs in the Renesas RX100 series deliver outstanding performance coupled with advanced power-saving features to better address the design requirements of applications that have limited power budgets. This white paper has highlighted the low-power modes these chips provide and explained some of the design options they offer for reducing current consumption. These impressive 32-bit devices give system engineers exciting opportunities to produce new products that stretch battery lifetimes to lengths previously impossible to achieve.

The current-consumption reduction techniques described herein facilitate the design of eco-friendly products that banish frequent battery replacements or recharges.

'True Low Power' is a trademark of Renesas Electronics America, Inc.

References and Resources

RX111 Group Datasheet:
http://documentation.renesas.com/doc/products/mpumcu/doc/rx_family/r01uh0365ej0100_rx111.pdf

RX100 User Hardware Manual:
http://www.am.renesas.com/rx100

RX100 Home Page (Americas):
http://www.am.renesas.com/rx100

RX100 Family Brochure:
http://www.am.renesas.com/media/products/mpumcu/rx/rx100/RX100Brochure.pdf

RX100 Customer Presentation:
http://www.am.renesas.com/media/products/mpumcu/rx/rx100/RX100SeriesOverview.pdf

RX100 Video Introduction:
https://www.youtube.com/watch?feature=player_embedded&v=lWrM0NOhtjA

RX100 Renesas Interactive Training Module:
http://www.renesasinteractive.com/course/view.php?id=553?loginActionInvoker=renesasinteractive