Application Note AN2091

RC5 Codec

By: Victor Kremin
Associated Project: Yes
Associated Part: CY8C26443

Summary
This Application Note discusses codec for the popular RC5-based remote control systems. The receiver codec reads remote system commands, decodes them and sends them as text messages via a RS232 interface. The transmitter receives RC5 commands from the serial interface, encodes them and sends them to the controlled equipment. The codec can be embedded into various control systems, PC program remote control, computer-based equipment control, etc.

Introduction
Various remote control systems are used in electronic equipment today. The RC5 control protocol is one of the most popular and is widely used to control numerous home appliances, entertainment systems and some industrial applications including utility consumption remote meter reading, contact-less apparatus control, telemetry data transmission, and car security systems. Philips originally invented this protocol and virtually all Philips' remotes use this protocol.

Following is a description of the RC5. When the user pushes a button on the hand-held remote, the device is activated and sends modulated infrared light to transmit the command. The remote separates command data into packets. Each data packet consists of a 14-bit data word, which is repeated if the user continues to push the remote button. The data packet structure is as follows:

- 2 start bits,
- 1 control bit,
- 5 address bits,
- 6 command bits.

The start bits are always logic '1' and intended to calibrate the optical receiver automatic gain-control loop. Next, is the control bit. This bit is inverted each time the user releases the remote button and is intended to differentiate situations when the user continues to hold the same button or presses it again.

The next 5 bits are the address bits and select the destination device. A number of devices can use RC5 at the same time. To exclude possible interference, each must use a different address. The 6 command bits describe the actual command. As a result, a RC5 transmitter can send the 2048 unique commands. The transmitter shifts the data word, applies Manchester encoding and passes the created one-bit sequence to a control carrier frequency signal amplitude modulator. The amplitude-modulated carrier signal is sent to the optical transmitter, which radiates the infrared light. In RC5 systems the carrier frequency has been set to 36 kHz. Figure 1 displays the RC5 protocol.

The receiver performs the reverse function. The photo detector converts optical transmission into electric signals, filters it and executes amplitude demodulation. The receiver output bit stream can be used to decode the RC5 data word. This operation is done by the microprocessor typically, but complete hardware implementations are present on the market as well. Single-die optical receivers are being mass produced by a number of companies such as Siemens, Temic, Sharp, Xiamen Hualian, Japanese Electric and others. Please note that the receiver output is inverted (log. 1 corresponds to illumination absence).

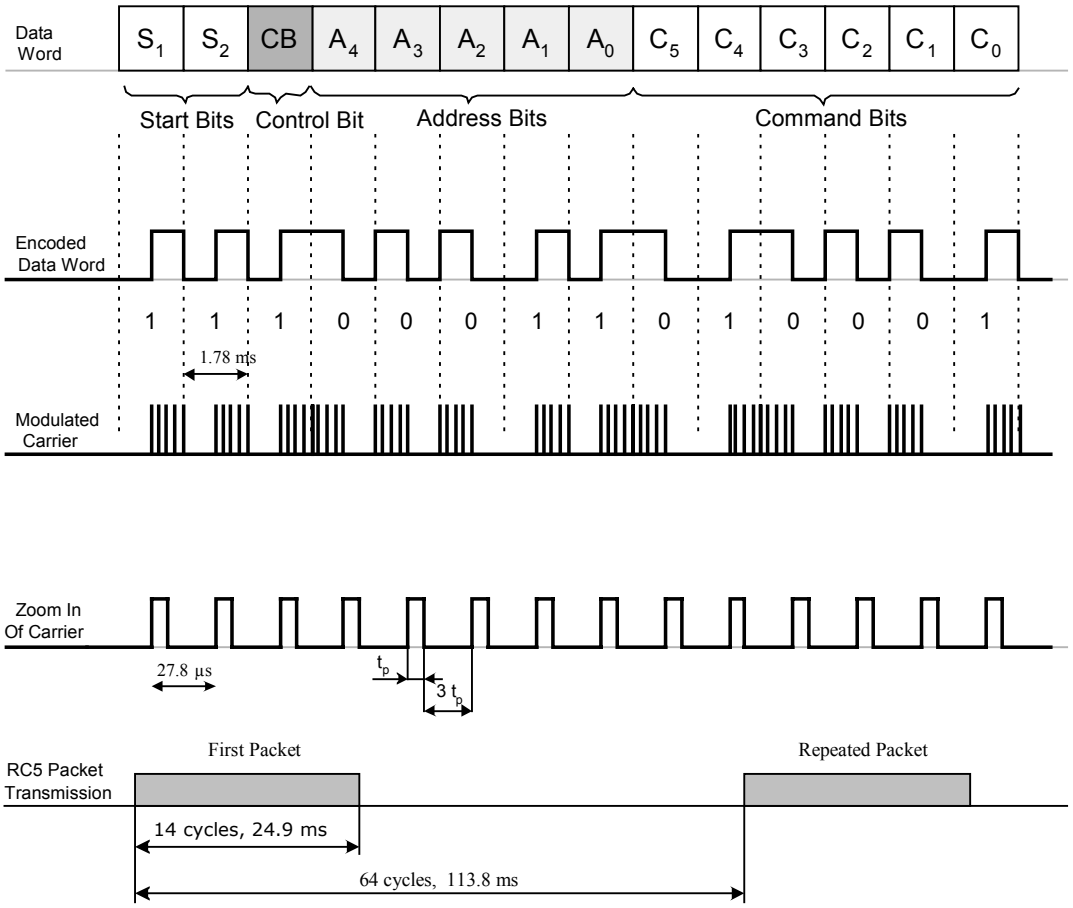

Figure 1: RC5 Protocol

The Codec Features

The codec has been built around the PSoC microcontroller together with an optical receiver and transmitter. The features' list of the codec is given below:

- Completely interrupt-driven design, no polling or wait cycles.
- Not CPU clock frequency dependent.
- The receiver operation is immune to interrupt latency up to 1 ms.
- All codec time intervals are obtained from a single 32 kHz watch crystal.
- Low CPU hardware resource utilization. The receiver consumes only one timer for measuring time intervals and the transmitter employs one timer to generate the carrier signal.
- Easy adaptation for processing of other remote system standards, both phase modulated and pulse-distance coded.

The Codec Schematic

Figure 2 illustrates the codec hardware. The schematic is quite simple. The infrared light reaches the receiver U_2 and is converted into electric pulses, which are processed by the CPU. MOSFET Q_1 drives the infrared LED D_1. Level translator U_3 forms the RS232 signal level for PC data exchange. The codec can be powered from a PC serial port, using an external low-dropout regulator, such as LP2950-5, because power consumption is low. In the PC port-powered design, light emission power must be reduced by increasing the value of R_2. The device can operate from 3.3 V supply, but U_2 must be operational at 3.3 V also. Note that the PSoC internal low-speed crystal oscillator is very sensitive to external noise.

That is why all the digital lines were routed far away from the oscillator pins. This must be taken into account during PCB layout as well.

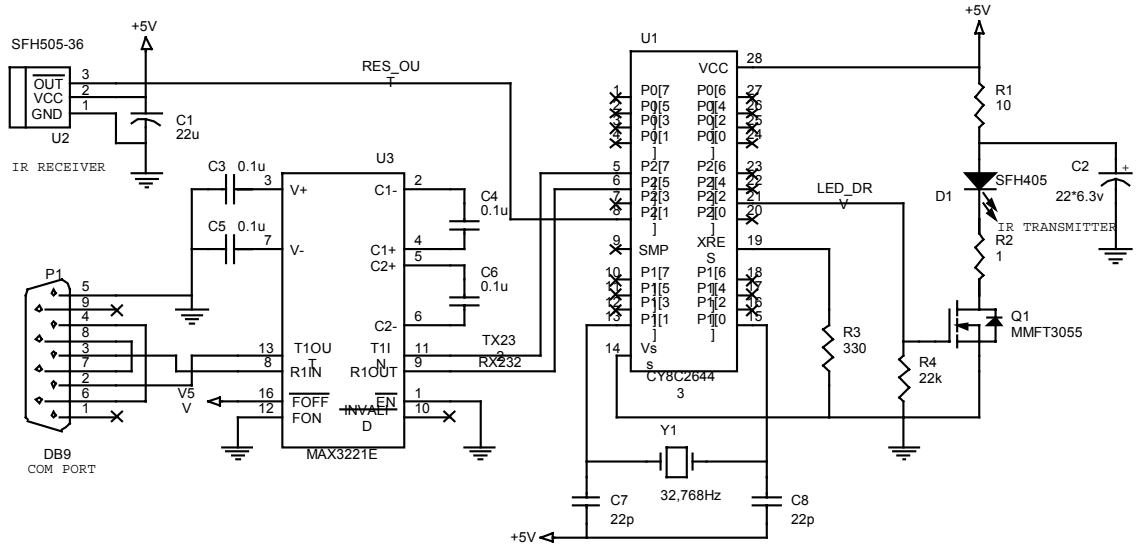

Figure 2: Codec Schematics

The PSoC Internals

Figure 3 illustrates the PSoC chip interconnect. The codec uses only digital modules. All analog modules are at the user's disposal. The version of this project, which uses an external photodiode instead of an integrated receiver, has been tested as well. The signal processing scheme was similar to Application Note AN2042 "Multifunctional Optical Sensor." That project utilizes most of the analog blocks, several additional digital blocks, and some external passive components. Moreover, the power consumption is noticeably larger.

As a result, this approach should be considered only if a multi-carrier receiver has to be designed. This will be described in a separate Application Note; multi-standard universal remote control with command-learning capabilities.

In Figure 3 the port labels in brackets display the corresponding port numbers, the italic font depicts the matching schematic net names and narrow lines have been used for presenting the clock lines. Gray color is used to mark the unused blocks, which can be used for implementation of additional features.

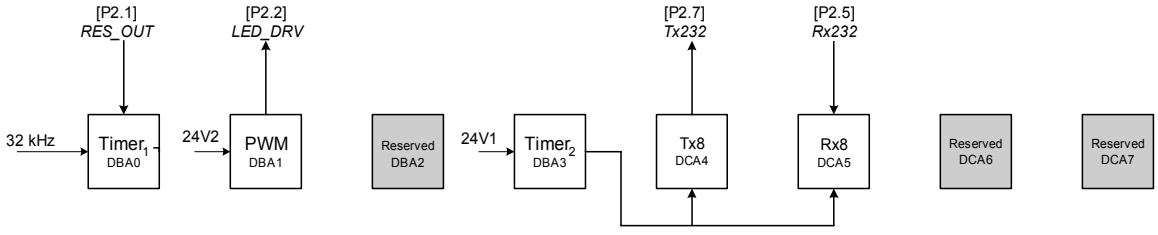

Figure 3: PSoC Digital User Module Placement

The codec supports two operational modes: receiver and transmitter. A full duplex mode implementation makes no sense because emitted transmitter light will reflect and be processed by the receiver.

The receiver will operate in loop mode, simply repeating the transmitted code. For decreasing resource utilization, codec hardware and software resources are shared between transmitter and receiver.

In receiver mode, $Timer_1$ is used for measuring the time intervals between adjacent pulses, using its capture capability. In addition, this timer simultaneously generates the timeouts. The timer capture mechanism is activated by each infrared receiver rising edge. It corresponds to the falling edge RC5 code pulse from Figure 1 because the infrared receiver inverts this signal. In transmitter mode, $Timer_1$ serves as the reference time generator to control the carrier frequency amplitude modulator. A pulse-width modulator (PWM) generates the carrier signal with a 25% duty cycle as described in the RC5 protocol standard. It is used simultaneously as the amplitude modulator.

$Timer_2$ is the baud rate timer for the UART, which is comprised of serial transmitter TX8 and receiver RX8.

The Codec Firmware

The codec firmware can be divided into two sections: receiver/transmitter and supplemental. Because the receiver and transmitter are completely interrupt driven, the primary code portion is implemented using state machines and placed in Interrupt Service Routines (ISR).

The receiver detection algorithm is based on measuring the time intervals between neighboring falling edges of the RC5 serial bit stream. The first bit is always considered equal to logic '1'. The following bits can be detected through analyzing the measured time intervals accordingly to the suggested equation set:

$$
\begin{aligned}
&\text{when } {}_{\Delta}\mathrm{t_j} \in \left[T_p\left(1-\alpha\right), T_p\left(1+\alpha\right) \right] \text{ then } b_{j+1} = b_j, \\
&\text{when } {}_{\Delta}\mathrm{t_j} \in \left[1.5 \cdot T_p\left(1-\alpha\right), 1.5 \cdot T_p\left(1+\alpha\right) \right] \text{ then} \\
&\qquad \text{if } b_j = 1 \text{ then } b_{j+1} = b_j, \; b_{j+2} = \text{not } b_j \\
&\qquad\qquad \text{else } b_{j+1} = \text{not } b_j \\
&\text{when } {}_{\Delta}\mathrm{t_j} \in \left[2 \cdot T_p\left(1-\alpha\right), 2 \cdot T_p\left(1+\alpha\right) \right] \text{ then } b_{j+1} = \text{not } b_j, \; b_{j+2} = b_j
\end{aligned}
$$

Equation (1)

where T_p is the bit transmission time in RC5 standard; its nominal value is 1.78 ms; ${}_{\Delta}\mathrm{t_j}$ is the measured time between the next two falling edges; α is the tolerance parameter; it is set to 0.05-0.12; b_j is the j^{th} bit in the RC5 data word.

These equations can be directly obtained via analysis of various bit-adjacent combinations depicted in Figure 1.

The receiver detection algorithm has been implemented as a state machine. Figure 4 illustrates the decoder states.

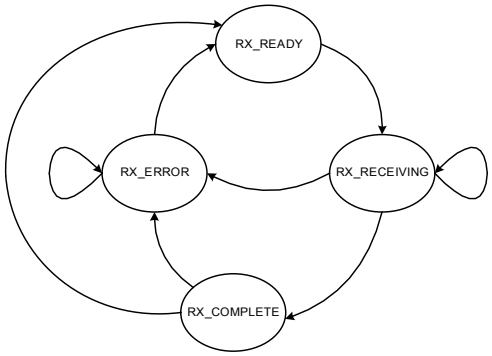

Figure 4: The Receiver State Diagram

Initially, the receiver is in the **RX_READY** state. If a falling edge is detected, the state is changed to **RX_RECEIVING**. The receiver remains in this state until all bits are received. If, during the bits' reception, it encounters any time interval that satisfies none of the equations from the set (1), or the reception process is stalled by a bits' absence within the predefined timeout, then the receiver is switched to **RX_ERROR** state. If all bits are received well and the second start bit is 1, the receiver is switched to **RX_COMPLETE** state.

The receiver preserves this state until decoded data is read by the CPU or the pause timeout has elapsed. During this timeout, no events may happen. If any crossings are detected, they shall be considered erroneous and shall change the receiver state to **RX_ERROR**. This state can be avoided only if no events happen during the predefined timeout. The transmission between receiver states is performed by $Timer_1$ and Global Input-Output ISRs. The $Timer_1$ period is set to 256 in this mode which corresponds to the interrupt period of 7.8 ms. This time tick is used to calculate all receiver timeouts. The RC5 detection algorithm was successfully tested in various conditions and showed complete absence of incorrectly detected RC5 codes.

The transmitter is implemented as a state machine as well. Figure 5 illustrates the transmitter operation:

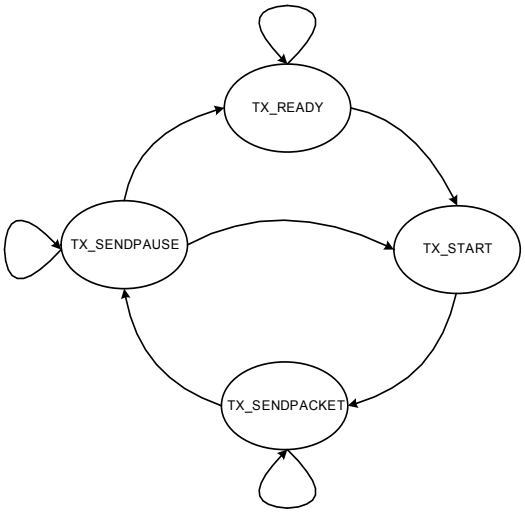

Figure 5: The Transmitter State Diagram

Initially, the transmitter is set to the **TX_READY** state. When the “start transmission” command is encountered, the transmitter goes to **TX_START** state. During this state, initialization of the internal variables is prepared. Next, an interrupt switches the receiver state to **TX_SENDPACKET**. The transmitter continues in this state until the RC5 packet is completely transmitted. After this, the receiver goes to **TX_SENDPAUSE** state for forming the pause between the packets. Typically, each command is repeated a predefined number of times. If a command must be repeated, the transmitter switches to **TX_START** again. In the opposite situation, there is a return to **TX_READY** state.

The Timer1 ISR performs transmission between transmitter states. $Timer_1$ is creates periodic interrupts with the period of 885 μs in transmitter mode. The timer period differentiates from the RC5 standard period less then 0.45%, which is completely suitable for remote code receivers. The carrier signal amplitude modulation is performed by switching the PWM unit with direct modification of the DBA1 Control register. To minimize the transmission jitter, which was caused by distinct ISR execution time, the PWM state is updated by the previously calculated value in the first line of the timer interrupt routine code. The PWM switching frequency is very close to the defined standard in the RC5 of 36 kHz and is determined by the PSoC high-speed oscillator accuracy. For some applications, the digital PLL can be used to refine PSoC high-speed oscillator frequency as described in the PSoC device data sheet.

Interface routines convert the decoded RC5 codes into text messages and send them via serial link. In addition, the text messages from the serial link are decoded and can be transmitted as RC5 commands. Incoming serial port data is processed by the corresponding ISR to eliminate serial port polling.

The supplemental software is responsible for peripheral set initialization and receiver mode changing. The main program is very simple: initialize the peripheral devices first, set codec mode as receiver and go to a perpetual loop. In this loop, we check the availability of the command string from the serial port. If the command has been received, it is parsed to extract the values of the RC5 data word. If no problems are encountered during this operation, we change the codec mode to transmitter and start command transmission. Next, we wait until the transmitter is busy and change the codec mode back to receiver after transmission is complete.

If the receiver has successfully decoded an RC5 command, we form the text message and send it via the serial port. Note that as described above, the RC5 codec is completely interrupt driven, so the processor can process other tasks and periodically check receiver or transmitter states for new decoded command availability or command transmission completion. Please note that interrupt latency in the transmitter mode causes bit-stream jitter and must be minimized.

Designers can utilize any terminal program to interact with the codec. If a message from the remote control has been successfully received, the codec sends the message in the following format:

"cb = v1, adr = v2, cmd = v3" where **v1** – control bit value 0..1, **v2** – address value 0..31, **v3** – command value, 0..63.

To send a command to the codec, the user must prepare and send a text string in the following format:

"sv1 av2 cv3 rv4" with a new line symbol ending, where **v1, v2, v3, v4** are values of control bit, address, command and packet repeats' number correspondingly. Please use small letters **s, a, c, r** to mark the numerical values. Photo 1 depicts a simple terminal program screenshot with received commands from two different remotes (from audio CD recorder and TV set). The default codec serial interface speed is set to 19200 baud without parity (8N1).

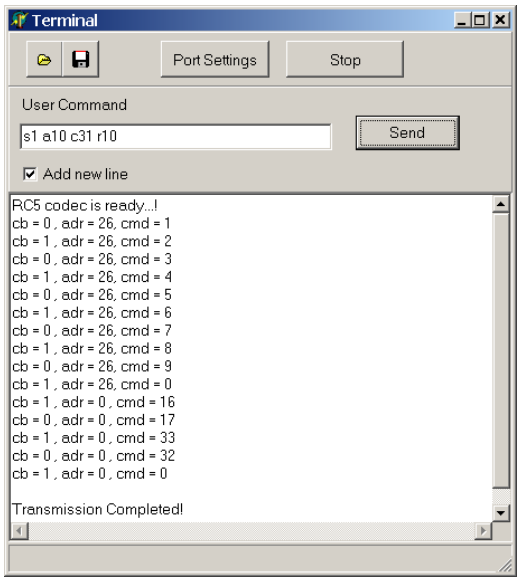

Photo 1: The Terminal Program

Codec Variances

The codec can easily be adopted for reception/transmission of alternative remote system standards. To process the alternative phase-manipulated codes, such as Nokia NRC17, the equation set (1) may remain unchanged, but T_p must be adjusted accordingly. The pulse-distance codes (such as from NEC, Sharp) reception algorithms are very simple because the command bit values are coded by different time intervals between adjacent pulses. Sometimes remote codes include the check sums or inverse command/address fields for better noise resistance. This checking must be added after packet reception. Some codes can use pulse-width modulation with constant bit transmission period. The reconfigurable PSoC architecture allows organizing decoding of such codes without any problems. The counter with enable input, which connects to the inverted receiver output, will permit users to accurately measure pulse width without CPU polling. The receiver ISR must only read the counter value and reset the counter to the next cycle for width measuring. Lastly, any PSoC processor can be used for codec implementation.

References

http://www.xs4all.nl/~sbp describes some remote control protocols, including RC5.

About the Author

Name: Victor Kremin
Title: Associate Professor
Background: Viktor had earned radiophysics diploma in 1996 from Ivan Franko National Lviv University, PhD degree in Computer Aided Design systems in 2000, and is presently working as Associate Professor at National University "Lvivska Politekhnika" (Ukraine).

His interests involve the full cycle of embedded systems design including various processors, operation systems and target applications.

Contact: vkremin@polynet.lviv.ua

Cypress MicroSystems, Inc.
22027 17th Avenue S.E. Suite 201
Bothell, WA 98021
Phone: 877.751.6100
Fax: 425.939.0999
http://www.cypressmicro.com/ / http://www.cypress.com/aboutus/sales_locations.cfm / support@cypressmicro.com