
AVR410: RC5 IR Remote Control Receiver on tinyAVR and megaAVR devices

APPLICATION NOTE

Introduction

Most audio and video systems are equipped with an infrared remote control. This application note describes a receiver for the frequently used Philips/Sony RC5 coding scheme.

Features

- Low-cost
- Compact design, only one external component
- Requires only one controller pin, any AVR® device can be used
- Size-efficient code
- Complementary of the Atmel® AVR415 RC5 IR Remote Control Transmitter on Atmel tinyAVR® and megaAVR® devices

Table of Contents

Introduction.....	1
Features.....	1
1. RC5 Coding Scheme.....	3
2. Timing.....	4
3. Software.....	5
4. Example Program.....	8
5. Revision History.....	12

1. RC5 Coding Scheme

The RC5 code is a 14-bit word bi-phase coded signal as seen in the figure below. The first two bits are start bits, always having the value “1”. The next bit is a control bit, which is toggled every time a button is pressed on the remote control transmitter. This gives an easy way of determining whether a button is pressed and held down, or pressed and released continuously.

Figure 1-1. RC5 Frame Format

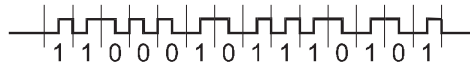


Five system bits hold the system address so that only the right system responds to the code. Usually, TV sets have the system address 0, VCRs the address 5, and so on. The command sequence is six bits long, allowing up to 64 different commands per address. The bits are transmitted in bi-phase code (also known as Manchester code) as shown below, along with an example where the command 0x35 is sent to system 5.

Figure 1-2. Bi-phase Coding

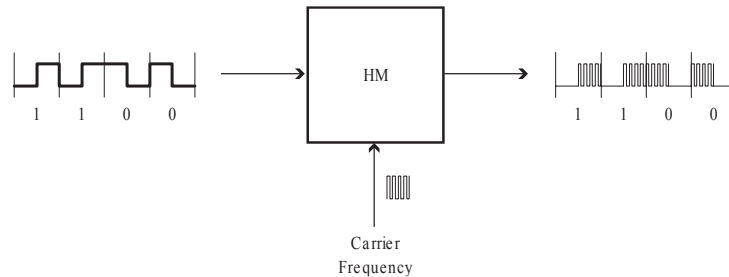


Figure 1-3. Example of Transmission



Note that the figures above show the signal that enters the Atmel ATtiny28 hardware modulator. The actual signal emitted by the IR-LED will be modulated with a certain carrier frequency as shown in the figure below.

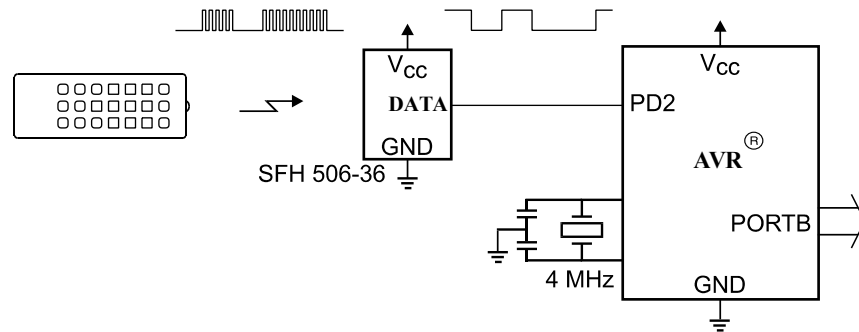
Figure 1-4. Signal Before and After Modulation



2. Timing

The bit length is approximately 1.8ms. The code is repeated every 114ms. To improve noise rejection, the pulses are modulated at 36kHz. This can be seen in the RC5 IR Receiver figure below. The easiest way to receive these pulses is to use an integrated IR-receiver/demodulator like the Siemens SFH 506-36. This is a 3-pin device that receives the infra-red burst and gives out the demodulated bit stream at the output pin. Note that the data is inverted compared to the transmitted data (i.e., the data is idle high). The output of the demodulator device is connected to PD2 on the AVR device. The decoded instructions in this case will be output on PORTB, but the chosen port is easily reconfigurable.

Figure 2-1. RC5 Receiver



3. Software

The assembly code found in AVR410.ASM contains the RC5 decode routine. In addition, it contains an example program which initializes the resources, decodes the RC5 data and outputs the received command on PORTB.

The Detect Subroutine

When the detect subroutine is called, it first waits for the data line to be idle high for more than 3.5ms. Then, a start bit can be detected. The length of the low part of the first start bit is measured. If no start bit is detected within 131ms, or if the low pulse is longer than 1.1ms, the routine returns indicating no command received.

The measurement of the start bit is used to calculate two reference times, ref1 and ref2, which are used when sampling the data line. The program uses the edge in the middle of every bit to synchronize the timing. 3/4 of a bit length after this edge, the line is sampled. This is in the middle of the first half of the next bit (see the figure below). The state is stored and the routine waits for the middle edge. Then, the timer is synchronized again and the steps are repeated for the following bits. If the synchronizing edge is not detected within 5/4 bit times from the previous synchronizing edge, this is detected as a fault and the routine terminates. When all the bits are received, the command and system address are stored in the "command" and "system" registers. The control bit is stored in bit 6 of "command".

Figure 3-1. Synchronizing and Sampling of the Data

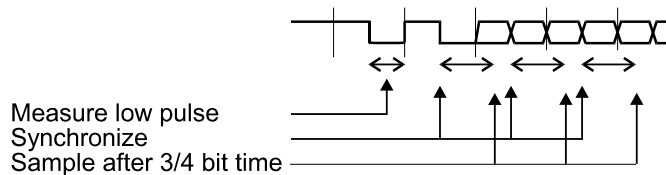


Table 3-1. "Decode" Subroutine Performance Figures

Parameter	Value
Code size	72 words
Register usage	Low registers used: 3 High registers used: 6 Global registers: 6 Pointers used: None

Table 3-2. "Detect" Register Usage

Register	Internal	Output
R1	"inttemp" - Used by TIM0_OVF	
R2	"ref1" - Holds timing information	
R3	"ref2" - Holds timing information	
R16	"temp" - Temporary Register	
R17	"timerL" - Timing Register	
R18	"timerH" - Timing Register	

Register	Internal	Output
R19		"system" - The System Address
R20		"command" - The Received Command
R21	"bitcnt" - Counts the bits received	

Timer/Counter 0 Overflow Interrupt Handler

The function of the timer interrupt is to generate a clock base for the timing required. The routine increments the "timerL" Register every 64µs, and the "timerH" every 16.384ms.

Table 3-3. "TIM0_OVF" Interrupt Handler Performance Figures

Parameter	Value
Code size	7 words
Execution cycles	6 + reti
Register usage	Low registers used: 2 High registers used: 2 Global registers: 0 Pointers used: None

Table 3-4. "TIM0_OVF" Register Usage

Register	Internal	Output
R0	"S" - Temporary Storage of Sreg	
R1	"inttemp" - Used by TIM0_OVF	
R17	"timerL" - Incremented every 64µs	
R18	"timerH" - Incremented every 16.384ms	

Example Program

The example program initializes the ports, sets up the timer, and enables interrupts. Then, the program enters an eternal loop, calling the detect routine. If the system address is correct, the command is output on PORTB.

Table 3-5. Overall Performance Figures

Parameter	Value
Code size	79 words - "detect" and "TIM0_OVF" 96 words - Complete example code
Register usage	Low registers: 4 High registers: 6 Pointers: None

Parameter	Value
Interrupt usage	Timer/Counter 0 Overflow Interrupt
Peripheral usage	Timer/Counter 0 Port D, pin 2 Port B (example program only)

4. Example Program

```
*****
;* APPLICATION NOTE FOR THE AVR FAMILY
;*
;* Number : AVR410
;* File Name : "rc5.asm"
;* Title : RC5 IR Remote Control Decoder
;* Date : 97.08.15
;* Version : 1.0
;* Support telephone : +47 72 88 43 88 (ATMEL Norway)
;* Support fax : +47 72 88 43 99 (ATMEL Norway)
;* Target MCU : AT90S1200
;*
;* DESCRIPTION
;* This Application note describes how to decode the frequently used
;* RC5 IR remote control protocol.
;*
;* The timing is adapted for 4 MHz crystal
;*
*****
.include "1200def.inc"
.device AT90S1200

.equ INPUT = 2 ; PD2
.equ SYS_ADDR = 0 ; The system address

.def S = R0 ; Storage for the Status Register
.def inttemp = R1 ; Temporary variable for ISR
.def ref1 = R2
.def ref2 = R3 ; References for timing

.def temp = R16 ; Temporary variable

.def timerL = R17 ; Timing variable updated every 14 us
.def timerH = R18 ; Timing variable updated every 16 ms
.def system = R19 ; Address data received
.def command = R20 ; Command received

.def bitcnt = R21 ; Counter

.cseg
.org 0
    rjmp reset
*****
;* "TIM0_OVF" - Timer/counter overflow interrupt handler
;*
;* The overflow interrupt increments the "timerL" and "timerH"
;* every 64us and 16,384us.
;*
;* Crystal Frequency is 4 MHz
;*
;* Number of words: 7
;* Number of cycles: 6 + reti
;* Low registers used: 1
;* High registers used: 3
;* Pointers used: 0
*****
.org OVFOaddr
TIM0_OVF:
    in S, sreg ; Store SREG
    inc timerL ; Updated every 64us
    inc inttemp
    brne TIM0_OVF_exit

    inc timerH ; if 256th int inc timer

TIM0_OVF_exit:
    out sreg, S ; Restore SREG
reti
*****
;* Example program
;*
;* Initializes timer, ports and interrupts.
;*
```



```

;* Calls "detect" in an endless loop and puts the result out on
;* port B.
;*
;* Number of words: 16
;* Low registers used: 0
;* High registers used: 3
;* Pointers used: 0
;*****
reset:
    ;ldi        temp,low(RAMEND)    ;Initialize stackpointer for parts with SW stack
    ;out
    ;ldi        temp,high(RAMEND)  ; Commented out since 1200 does not have SRAM
    ;out
    ;SPH,temp

    ldi        temp,1              ;Timer/Counter 0 clocked at CK
    out        TCCR0,temp

    ldi        temp,1<<TOIE0      ;Enable Timer0 overflow interrupt
    out        TIMSK,temp

    ser        temp                ; PORTB as output
    out        DDRB,temp
    sei
                                ;Enable global interrupt

main:
    rcall     detect              ;Call RC5 detect routine

    cpi      system,SYS_ADDR      ;Responds only at the specified address
    brne     release

    andi     command,0x3F         ;Remove control bit
    out     PORTB,command

    rjmp     main

release:
    clr      command              ;Clear PORTB
    out     PORTB,command
    rjmp     main

;*****
;* "detect" - RC5 decode routine
;*
;* This subroutine decodes the RC5 bit stream applied on PORTD
;* pin "INPUT".
;*
;* If success: The command and system address are
;* returned in "command" and "system".
;* Bit 6 of "command" holds the toggle bit.
;*
;* If failed: $FF in both "system" and "command"
;*
;* Crystal frequency is 4MHz
;*
;* Number of words:72
;* Low registers used: 3
;* High registers used: 6
;* Pointers used: 0
;*****
detect:
    clr      inttemp              ; Init Counters
    clr      timerH

detect1:
    clr      timerL

detect2:
    cpi      timerH,8             ;If line not idle within 131ms
    brlo    dll
    rjmp     fault                ;then exit

dll:
    cpi      timerL,55            ;If line low for 3.5ms
    brge    start1               ;then wait for start bit

    sbis    PIND,INPUT            ;If line is
    rjmp    detect1               ;low - jump to detect1
    rjmp    detect2               ;high - jump to detect2

```

```

start1:
    cpi        timerH,8           ;If no start bit detected
    brge      fault             ;within 130ms then exit

    sbic      PIND,INPUT        ;Wait for start bit
    rjmp     start1

    clr      timerL             ;Measure length of start bit

start2:
    cpi        timerL,17         ;If startbit longer than 1.1ms,
    brge      fault             ;exit

    sbis      PIND,INPUT        ;Positive edge of 1st start bit
    rjmp     start2

    mov      temp,timerL        ;timer is 1/2 bit time
    clr      timerL

    mov      ref1,temp
    lsr      ref1
    mov      ref2,ref1
    add      ref1,temp          ;ref1 = 3/4 bit time
    lsl      temp
    add      ref2,temp          ;ref2 = 5/4 bit time

start3:
    cp        timerL,ref1        ;If high period St2 > 3/4 bit time
    brge      fault             ;exit

    sbic      PIND,INPUT        ;Wait for falling edge start bit 2
    rjmp     start3
    clr      timerL
    ldi      bitcnt,12          ;Receive 12 bits
    clr      command
    clr      system

sample:
    cp        timerL,ref1        ;Sample INPUT at 1/4 bit time
    brlo     sample

    sbic      PIND,INPUT        ;Jump if line high
    rjmp     bit_is_a_1

bit_is_a_0:
    clc                                ;Store a '0'
    rol      command
    rol      system

                                ;Synchronize timing
bit_is_a_0a:
    cp        timerL,ref2        ;If no edge within 3/4 bit time
    brge      fault             ;exit
    sbis      PIND,INPUT        ;Wait for rising edge
    rjmp     bit_is_a_0a        ;in the middle of the bit

    clr      timerL
    rjmp     nextbit

bit_is_a_1:
    sec                                ;Store a '1'
    rol      command
    rol      system

                                ;Synchronize timing
bit_is_a_1a:
    cp        timerL,ref2        ;If no edge within 3/4 bit time
    brge      fault             ;exit
    sbic      PIND,INPUT        ;Wait for falling edge
    rjmp     bit_is_a_1a        ;in the middle of the bit

    clr      timerL

nextbit:
    dec      bitcnt              ;If bitcnt > 0
    brne     sample             ;get next bit
                                ;All bits successfully received!
    mov      temp,command        ;Place system bits in "system"
    rol      temp

```

```

rol    system
rol    temp
rol    system

bst    system,5        ;Move toggle bit
bld    command,6       ;to "command"
                        ;Clear remaining bits
andi   command,0b01111111
andi   system,0x1F

ret

fault:
ser    command        ;Both "command" and "system"
ser    system         ;0xFF indicates failure
ret

```

5. Revision History

Doc Rev.	Date	Comments
1473C	08/2016	New template and some minor changes
1473B	05/2002	
1473A		Initial document release

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, AVR®, tinyAVR®, megaAVR®, and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. Other terms and product names may be trademarks of others.

DISCLAIMER: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death ("Safety-Critical Applications") without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.