AVR410: RC5 IR Remote Control Receiver

8-bit AVR® Microcontroller

Application Note

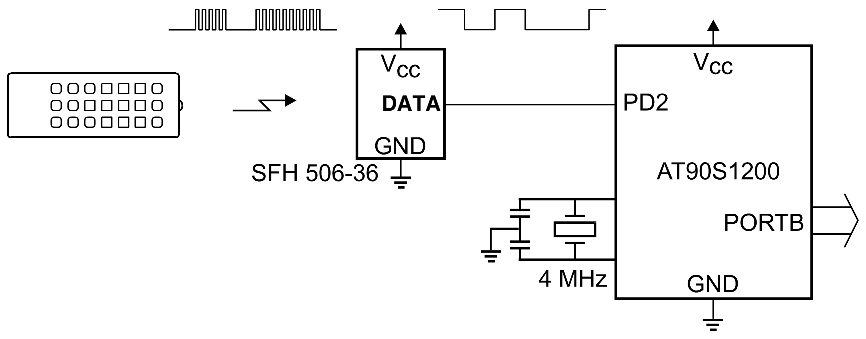
Features

- Low-cost
- Compact Design, Only One External Component
- Requires Only One Controller Pin, Any AVR Device Can be Used
- Size-efficient Code

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.

Figure 1. RC5 Receiver



The RC5 code is a 14-bit word bi-phase coded signal (See Figure 2). The two first bits are start bits, always having the value one. The next bit is a control bit or toggle bit, which is inverted every time a button is pressed on the remote control transmitter. 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 in Figure 3. An example where the command 0x35 is sent to system 5 is shown in Figure 4.

Rev. 1473B–AVR–05/02

Figure 2. RC5 Frame Format

St_1	St_2	Ctrl	S_4	S_3	S_2	S_1	S_0	C_5	C_4	C_3	C_2	C_1	C_0

Figure 3. Bi-phase Coding

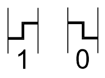
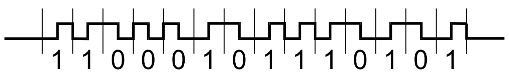


Figure 4. Example of Transmission



Timing

The bit length is approximately 1.8 ms. The code is repeated every 114 ms. To improve noise rejection, the pulses are modulated at 36 kHz. 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).

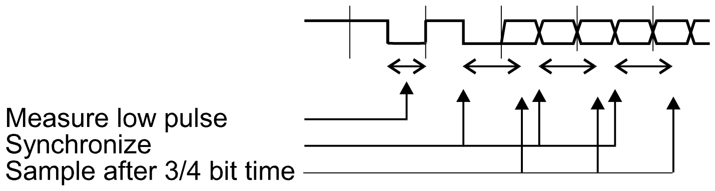
The Software

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 port B.

The Detect Subroutine

When the detect subroutine is called, it first waits for the data line to be idle high for more than 3.5 ms. 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 131 ms, or if the low pulse is longer than 1.1 ms, the routine returns indicating no command received.

Figure 5. Synchronizing and Sampling of the Data



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 bit length after this edge, the line is sampled. This is in the middle of the first half of the next bit (see Figure 5). The state is stored and the routine waits for the middle edge. Then, the timer is synchronized again and everything is 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".

Table 1. "Decode" Subroutine Performance Figures

Parameter	Value
Code Size	72 words
Execution Cycles	
Register Usage	Low Registers Used: 3 High Registers Used: 6 Global Registers: 6 Pointers Used: None

Table 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	
R19		"system"– The System Address
R20		"command" – The Received Command
R21	"bitcnt" – Counts the Bits Received	

Timer/Counter0 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 64 µs, and the "timerH" every 16,384 ms.

Table 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 4. "TIM0_OVF" Register Usage

Register	Internal	Output
R0	"S" – Temporary Storage of Sreg	
R1	"inttemp" – Used by TIM0_OVF	
R17	"timerL" – Incremented every 64 µs	
R18	"timerH" – Incremented every 16,384 ms	

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 port B.

Table 5. Overall Performance Figures

Parameter	Value
Code Size	79 words – “detect” and “TIM0_OVF 96 words – Complete Application Note
Register Usage	Low Registers: 4 High Registers: 6 Pointers: None
Interrupt Usage	Timer/Counter 0 Interrupt
Peripheral Usage	Timer/Counter Port D, pin 2 Port B (example program only)

```
;***************************************************************************
;* A P P L I C A T I O N   N O T E   F O R   T H E   A V R   F A M I L Y
;*
;* 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

.equu   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           ; Reference 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 OVF0addr
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      SPL,temp
    ;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     dl1
    rjmp     fault              ;then exit

dl1:
    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
    rjmp     start2              ;Positive edge of 1st start bit

    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
    rjmp     bit_is_a_1          ;Jump if line high

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 sucessfully 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
```

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 487-2600

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 2-40-18-18-18
FAX (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-76-58-30-00
FAX (33) 4-76-58-34-80

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

1473B–AVR–05/02 0M