AVR410: RC5 IR Remote Control Receiver

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.
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.

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”.

Figure 2. RC5 Frame Format

```
St. St. Ctrl S1 S2 S3 S4 S5 C1 C2 C3 C4 C5
```

Figure 3. Bi-phase Coding

```
0 1
```

Figure 4. Example of Transmission

```
1 1 0 0 0 1 1 1 1 0 1 1
```

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”.

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 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.

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.

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”.
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 1. “Decode” Subroutine Performance Figures

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Size</td>
<td>72 words</td>
</tr>
<tr>
<td>Execution Cycles</td>
<td></td>
</tr>
<tr>
<td>Register Usage</td>
<td>Low Registers Used: 3</td>
</tr>
<tr>
<td></td>
<td>High Registers Used: 6</td>
</tr>
<tr>
<td></td>
<td>Global Registers: 6</td>
</tr>
<tr>
<td></td>
<td>Pointers Used: None</td>
</tr>
</tbody>
</table>

Table 2. “Detect” Register Usage

<table>
<thead>
<tr>
<th>Register</th>
<th>Internal</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>“inttemp” – Used by TIM0_OVF</td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>“ref1” – Holds Timing Information</td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td>“ref2” – Holds Timing Information</td>
<td></td>
</tr>
<tr>
<td>R16</td>
<td>“temp” – Temporary Register</td>
<td></td>
</tr>
<tr>
<td>R17</td>
<td>“timerL” – Timing Register</td>
<td></td>
</tr>
<tr>
<td>R18</td>
<td>“timerH” – Timing Register</td>
<td></td>
</tr>
<tr>
<td>R19</td>
<td>“system” – The System Address</td>
<td></td>
</tr>
<tr>
<td>R20</td>
<td>“command” – The Received Command</td>
<td></td>
</tr>
<tr>
<td>R21</td>
<td>“bitcnt” – Counts the Bits Received</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. “TIM0_OVF” Interrupt Handler Performance Figures

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Size</td>
<td>7 words</td>
</tr>
<tr>
<td>Execution Cycles</td>
<td>6 + reti</td>
</tr>
<tr>
<td>Register Usage</td>
<td>Low Registers Used: 2</td>
</tr>
<tr>
<td></td>
<td>High Registers Used: 2</td>
</tr>
<tr>
<td></td>
<td>Global Registers: 0</td>
</tr>
<tr>
<td></td>
<td>Pointers Used: None</td>
</tr>
</tbody>
</table>

Table 4. “TIM0_OVF” Register Usage

<table>
<thead>
<tr>
<th>Register</th>
<th>Internal</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>“S” – Temporary Storage of Sreg</td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>“inttemp” – Used by TIM0_OVF</td>
<td></td>
</tr>
<tr>
<td>R17</td>
<td>“timerL” – Incremented every 64 µs</td>
<td></td>
</tr>
<tr>
<td>R18</td>
<td>“timerH” – Incremented every 16,384 ms</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Size</td>
<td>79 words – “detect” and “TIM0_OVF”</td>
</tr>
<tr>
<td></td>
<td>96 words – Complete Application Note</td>
</tr>
<tr>
<td>Register Usage</td>
<td>Low Registers: 4</td>
</tr>
<tr>
<td></td>
<td>High Registers: 6</td>
</tr>
<tr>
<td></td>
<td>Pointers: None</td>
</tr>
<tr>
<td>Interrupt Usage</td>
<td>Timer/Counter 0 Interrupt</td>
</tr>
<tr>
<td>Peripheral Usage</td>
<td>Timer/Counter</td>
</tr>
<tr>
<td></td>
<td>Port D, pin 2</td>
</tr>
<tr>
<td></td>
<td>Port B (example program only)</td>
</tr>
</tbody>
</table>
;***************************************************************************
;* 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 ; 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 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
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
lsr   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
mov temp,command ;All bits successfully received!
rol temp
rol system
rol temp
rol system

bet 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 Corporation 2002.
Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company’s standard warranty which is detailed in Atmel’s Terms and Conditions located on the Company’s web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel’s products are not authorized for use as critical components in life support devices or systems.

ATMEL® and AVR® are the registered trademarks of Atmel.

Other terms and product names may be the trademarks of others.