Microcontroller systems and Development
Ex: Toshiba TMP87CH29U

TMP87CH29U/N, TMP87CK29U/N, TMP87CM29U/N

The TMP87CH29/K29/M29 are high-speed and high-performance 8-bit single chip microcomputers. These MCU contains CPU core, ROM, RAM, a LCD driver, multi-function timer/counters, an AD converter, two clock generators and a serial interface (UART) on a chip.

<table>
<thead>
<tr>
<th>Part No.</th>
<th>ROM</th>
<th>RAM</th>
<th>Package</th>
<th>OTP_MCU</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMP87CH29U</td>
<td>16 K x 8-bit</td>
<td></td>
<td>P-LQFP64-1010-0.50D</td>
<td>TMP87PM29U</td>
</tr>
<tr>
<td>TMP87CH29N</td>
<td></td>
<td>1 K x 8-bit</td>
<td>P-SDIP64-750-1.78</td>
<td>TMP87PM29N</td>
</tr>
<tr>
<td>TMP87CK29U</td>
<td>24 K x 8-bit</td>
<td></td>
<td>P-LQFP64-1010-0.50D</td>
<td>TMP87PM29U</td>
</tr>
<tr>
<td>TMP87CK29N</td>
<td></td>
<td></td>
<td>P-SDIP64-750-1.78</td>
<td>TMP87PM29N</td>
</tr>
<tr>
<td>TMP87CN29U</td>
<td>32 K x 8-bit</td>
<td></td>
<td>P-LQFP64-1010-0.50D</td>
<td>TMP87PM29U</td>
</tr>
<tr>
<td>TMP87CN29N</td>
<td></td>
<td></td>
<td>P-SDIP64-750-1.78</td>
<td>TMP87PM29N</td>
</tr>
</tbody>
</table>

Features

◆ 8-bit single chip microcomputer TLCS-870 Series
◆ Instruction execution time: 0.5 μs (at 8 MHz), 122 μs (at 32.768 kHz)
◆ 412 basic instructions
  - Multiplication and Division (8 bits x 8 bits, 16 bits ÷ 8 bits)
  - Bit manipulations (set/clear/complement/move/test/exclusive or)
  - 16-bit data operations
  - 1-byte jump/subroutine-call (Short relative jump/Vector call)
◆ 13 interrupt sources (External: 4, Internal: 9)
  - All sources have independent latches each, and nested interrupt control is available.
  - 2 edge-selectable external interrupts with noise reject
  - High-speed task switching by register bank changeover
◆ 7 Input/Output ports (43 pins)
  - High current output: 3 pins (typ. 20 mA)
◆ 18-bit Timer/Counter
  - Timer, Event counter, Pulse width measurement, Frequency measurement modes
Ex: Toshiba TMP87CH29U

◆ Four 8-Bit Timer/Counters
  ● Timer, Event counter, Capture (Pulse width/duty measurement), PWM output, Programmable divider output modes

◆ Time Base Timer (Interrupt frequency: 1 Hz to 16384 Hz)

◆ Divider output function (frequency: 1 kHz to 8 kHz)

◆ Watchdog Timer
  ● Interrupt source/reset output (programmable)

◆ Universal asynchronous receiver and transmitter (UART)
  ● With 8-bit transmit/receive data buffer
  ● Transfer clock, Select of with/without parity bit.

◆ LCD driver/Controller
  ● LCD direct drive capability (max. 12-digit display at 1/4 duty LCD).
  ● 1/4, 1/3, 1/2 duties or static drive are programmably selectable.
  ● With display memory.

◆ Dual clock operation
  ● Single/Dual-clock mode (option)

◆ Five power saving operating modes
  ● SLOW mode: Low power consumption operation using low-frequency clock (32.768 kHz).
  ● IDLE1 mode: CPU stops, and peripherals operate using high-frequency clock. Release by interrupts.
  ● IDLE2 mode: CPU stops, and peripherals operate using high and low frequency clock. Release by interrupts.

◆ Wide operating voltage: 2.7 to 5.5V at 4.19 MHz/32.768 kHz, 4.5 to 5.5 V at 8 MHz/32.768 kHz

◆ Emulation Pod: BM87CM29U0A
Ex: Toshiba TMP87CH29U

Block Diagram
Ex: Sphygmomanometer

An instrument for measuring blood pressure, particularly in arteries.

Digital with manual or automatic inflation. These are electronic, easy to operate and practical in noisy environments. They measure mean arterial pressure (MAP) and use algorithms to calculate systolic and diastolic values. In this sense, they do not actually measure the blood pressure, but derive the readings.

An oscillometric type electronic sphygmomanometer which determines the blood pressure by the steps of compressing an artery with an inflated cuff, detecting pulse wave amplitude while the inflated cuff being deflated and comparing the pressure provided by the cuff on the artery with the pulse wave amplitude. The cuff pressure corresponding to the maximum pulse wave amplitude is determined as the mean the blood pressure, the cuff pressure corresponding to the pulse wave amplitude of the high pressure side which is equivalent to 50% of the maximum pulse wave amplitude as the systolic blood pressure, the cuff pressure corresponding to the pulse wave amplitude of the low pressure side which is equivalent to 70% of the maximum pulse wave amplitude as the diastolic blood pressure.
Ex: Sphygmomanometer - top

- Electric release valve
- Electric pump
- To cuff
- Pressure sensor
- LCD display
- Controls
Ex: Sphygmomanometer - bottom

Toshiba microcontroller
Timer/Alarm System development
The Scenix SX family of configurable communications controllers are fabricated in an advanced CMOS process technology. The advanced process, combined with a RISC-based architecture, allows high-speed computation, flexible I/O control, and efficient data manipulation. Throughput is enhanced by operating the device at frequencies up to 50/75 MHz and by optimizing the instruction set to include mostly single-cycle instructions. In addition, the SX architecture is deterministic and totally reprogrammable. The unique combination of these characteristics enables the device to implement hard real-time functions as software modules (Virtual Peripheral™) to replace traditional hardware functions.

On-chip functions include a general-purpose 8-bit timer with prescaler, an analog comparator, a brown-out detector, a watchdog timer, a power-save mode with multi-source wakeup capability, an internal R/C oscillator, user-selectable clock modes, and high-current outputs.
System schematic
Program Flowchart

START

Configure: Port direction, Port logic levels, Port voltage levels

Sense START button

N
Button ?

Y

20 ms debounce

sense button pin again

N
Button ?

Y

Read Port B - Load Regs with timer values

Turn on Yellow LED, Generate On Beep

Sense clock bit on Port

A

Bit = Hi ?

N
Clear clock Flag

Y

Check clock Flag

Y
Flag already set?

N

Set clock Flag - Decrement timer regs

C

B

A
Program Flowchart

C

A

N
Time
regs
=0?
Y

Turn off Yellow LED

Read Port C - Load
regs with timer values

Turn on Red LED -
Generate OFF Beep

B

Sense clock bit on Port
A

Bit = Hi 
N
Y

Clear clock Flag

Check clock Flag

Flag already
set?
N
Y

Set clock Flag -
Decrement timer regs

Timer
regs
=0?
N
Y

Turn Off Red Led
# System memory map

<table>
<thead>
<tr>
<th>Port</th>
<th>Function</th>
<th>Reg.</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA0</td>
<td>1PPS (IN)</td>
<td>$0F</td>
<td>Port Direction (1 = in, 0 = out)</td>
</tr>
<tr>
<td>RA1</td>
<td>Yellow LED (OUT)</td>
<td>$0E</td>
<td>Pullup resistors (1 = none, 0 = pullup)</td>
</tr>
<tr>
<td>RA2</td>
<td>Red LED (OUT)</td>
<td>$0D</td>
<td>Logic Levels (1 = TTL, 0 = CMOS)</td>
</tr>
<tr>
<td>RA3</td>
<td>Start Button (IN)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RB0</td>
<td>10 Minute SW1 (IN)</td>
<td>$09</td>
<td>10 minute timer values</td>
</tr>
<tr>
<td>RB1</td>
<td>20 Minute SW1 (IN)</td>
<td>$05</td>
<td></td>
</tr>
<tr>
<td>RB2</td>
<td>30 Minute SW1 (IN)</td>
<td>$0B</td>
<td>20 minute timer values</td>
</tr>
<tr>
<td>RB3</td>
<td>40 Minute SW1 (IN)</td>
<td>$08</td>
<td></td>
</tr>
<tr>
<td>RB4</td>
<td>50 Minute SW1 (IN)</td>
<td>$08</td>
<td>30 minute timer values</td>
</tr>
<tr>
<td>RB5</td>
<td>60 Minute SW1 (IN)</td>
<td>$0A</td>
<td></td>
</tr>
<tr>
<td>RB6</td>
<td>3 Minute Timer SW3 (IN)</td>
<td>$09</td>
<td>40 minute timer values</td>
</tr>
<tr>
<td>RB7</td>
<td>PCM speaker (OUT)</td>
<td>$0C</td>
<td></td>
</tr>
<tr>
<td>RC0</td>
<td>10 Minute SW2 (IN)</td>
<td>$0F</td>
<td>50 minute timer values</td>
</tr>
<tr>
<td>RC1</td>
<td>20 Minute SW2 (IN)</td>
<td>$10</td>
<td></td>
</tr>
<tr>
<td>RC2</td>
<td>30 Minute SW2 (IN)</td>
<td></td>
<td>60 minute timer values</td>
</tr>
<tr>
<td>RC3</td>
<td>40 Minute SW2 (IN)</td>
<td>$0E.2</td>
<td>30 sec. beep flag</td>
</tr>
<tr>
<td>RC4</td>
<td>50 Minute SW2 (IN)</td>
<td>$0E.1</td>
<td>3 min. sw. beep flag</td>
</tr>
<tr>
<td>RC5</td>
<td>60 Minute SW2 (IN)</td>
<td>$0E 0</td>
<td>1 sec. Flag</td>
</tr>
<tr>
<td>RC6</td>
<td>N.C.</td>
<td>$12</td>
<td>3 sec. Beep counter</td>
</tr>
<tr>
<td>RC7</td>
<td>N.C.</td>
<td>$11</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$10</td>
<td>Variable delays for beeps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$13</td>
<td>$B4 = 180 secs; 3 min. ctr. Reg.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$15</td>
<td>$02 = 2 sec. Beep ctr.</td>
</tr>
</tbody>
</table>
ORG $0

DEVICE SX28L, OSC1MHZ, TURBO
RESET Start
ID 'punchual'
FREQ 1_000_000
STACKX ; expand stack to 8 levels

j330 JMP @_330 ; set up page jump

ORG $50

; --------------------------------------
; port direction config.
; --------------------------------------
Start MODE $0F
MOV !ra, #%1001
MOV !rb, #%01111111
MOV !rc, #%00111111

; --------------------------------------
; port startup logic levels
; --------------------------------------
MOV ra, #%1000
MOV rb, #%00000000
MOV rc, #%00000000
MODE $0E
MOV !ra, #%0111
MOV !rb, #%10000000
MOV !rc, #%11000000
MODE $0D
MOV !ra, #%1110
MOV !rb, #%11111111
MOV !rc, #%11111111

; --------------------------------------
; read and debounce START button
; --------------------------------------
butn MOV W, RA
SNE RA.3
JMP butn

ms20 CLR $0C
MOV $0D, #20
Loop DECSZ $0C
JMP Loop
DECSZ $0D
JMP Loop

butn2 MOV W, RA
SNE RA.3
JMP butn

; --------------------------------------
; read time ON switches
; --------------------------------------
ton MOV W, RB
JNE RB.0, ten

; --------------------------------------
; read port A again
; button still pressed?
; retry if not

; --------------------------------------
; read port B
; 10 min
Assembly code

JNB RB.1, twen ; 20 min
JNB RB.2, thir ; 30 min
JNB RB.3, four ; 40 min
JNB RB.4, fif ; 50 min
JNB RB.5, six ; 60 min
ten CALL ldten
JMP countsec
twen CALL ldtwen
JMP countsec
thir CALL ldthir
JMP countsec
four CALL ldfour
JMP countsec
fif CALL ldfif
JMP countsec
six CALL ldsix
JMP countsec

;*******************************
;load regs with timer values
;*******************************
ldten MOV $08, #03 ; 0258 hex = 600 sec
MOV $09, #58
RET
ldtwen MOV $08, #05 ; 04B0 hex = 1200 sec
MOV $09, #B0
RET
ldthir MOV $08, #08 ; 0708 hex = 1800 sec
MOV $09, #88
RET
ldfour MOV $08, #0A ; 0960 hex = 2400 sec
MOV $09, #60
RET
ldfif MOV $08, #0C ; 0BB8 hex = 3000 sec
MOV $09, #B8
RET
ldsix MOV $08, #0F ; 0E10 hex = 3600 sec
MOV $09, #10
RET

;*******************************
;wait for ON-time timeout
;*******************************
countsec SETB RA.1 ; turn on YELLOW LED
call hbeep ;alert tone
rdclk MOV W, RA
CALL @j330 ; read clock in bit
JB RA.0, chflag ; check for 3 min 30 sec switch
CLR B ; see if flag already set
JMP rdclk ; if port=lo, clear flag
chflag JB $0E.0, rdclk ; wait for next rising edge
SETB $0E.0 ; if flag already set, wait
SETB $0E.0 ; else, set flag

;*******************************
;count decrement routine
;*******************************
DJNZ $09, rdclk ; LSR = 0?
DJNZ $08, rdclk ; MSR = 0?
CLRB RA.1 ; time up - turn off YELLOW LED

; *******************************************************
; read time OFF switches
; *******************************************************
toff MOV W, RC
JNB RC.0, Cten
JNB RC.1, Cten
JNB RC.2, Cthir
JNB RC.3, Cfour
JNB RC.4, Cfif
JNB RC.5, Csix

Cten CALL ldtten
JMP countsec2

Ctwen CALL ldtwen
JMP countsec2

Cthir CALL ldthir
JMP countsec2

Cfour CALL ldfour
JMP countsec2

Cfif CALL ldfif
JMP countsec2

Csix CALL ldsix
JMP countsec2

; *******************************************************
; wait for ON-time timeout
; *******************************************************
countsec2

SETB RA.2 ; turn on RED LED
CALL lbeep ; alert tone

rdclk2 MOV W, RA
JB RA.0, chflag2 ; read clock in bit
CLR $0E.0 ; see if flag already set
JMP rdclk2

chflag2 JB $0E.0, rdclk2 ; if port=lo, clear flag
SET $0E.0 ; wait for next rising edge

; *******************************************************
; count decremented routine
; *******************************************************
DJNZ $09, rdclk2
DJNZ $08, rdclk2

CLR $0E.0 ; time up - turn off RED LED
JMP ton ; repeat cycle

; *******************************************************
; beep subroutines
; *******************************************************
hbeep MOV $12, #3 ; set for 3 seconds
ml MOV W, RA
JB RA.0, flg ; read clock port
CLR $0E.0 ; flag already set?

SETB RB.7 ; port = lo, clear flag
CALL delay ; set sound bit

; wait
CLRB  RB.7 ; clr sound bit
CALL  delay ; wait
JMP   ml
flg   JB  $0E.0, m1 ; flag already set?
SETB $0E.0 ; else set flag
DJNZ  $12, m1 ; more sound bits
RET

delay  CLR  $10 ; variable delay routine
MOV   $11, #4
nloop DJNZ $10, nloop
RET

lbeep MOV  $12, #3 ; set for 3 seconds
MOV   W, RA ; read clock port
JB    RA.0, flg2 ; flag already set?
CLRB $0E.0 ; port = 10, clear flag
SETB RB.7 ; set sound bit
CALL  delay2 ; wait
CLRB RB.7 ; clr sound bit
CALL  delay2 ; wait
JMP   m2

flg2 JB  $0E.0, m2 ; flag already set?
SETB $0E.0 ; else set flag
DJNZ  $12, m2 ; more sound bits
RET

delay2 CLR  $10 ; variable delay routine
MOV   $11, #4
lp    DJNZ $10, lp
DJNZ  $11, lp
RET

; 3 min-on/ 30 sec-off special timer
; *********************
_330   MOV  $13, #$B4 ; 180 secs for 3 minutes
MOV   $14, #$1E ; 30 secs
CLRB $0E.1 ; clear 3minbeep flag
CLRB $0E.2 ; clear 30 sec beep flag
JNB   RB.6, BACK ; timer switch set? No, go back.
JNB   $0E.1, D3MIN ; 3min beep flag set? No, dece 3min counter
SETB $0E.1 ; else set beep flag

; start 3minute beep
; *********************
m3    MOV  $15, #2 ; set for 2 seconds
MOV   W, RA ; read clock port
JB    RA.0, flg3 ; flag already set?
CLRB $0E.0 ; port = 10, clear flag
SETB RB.7 ; set sound bit

; variable delay
Assembly code

CLR $10
MOV $11, #2
;variable delay routine

oloop DJNZ $10, oloop
DJNZ $11, oloop

;************************
;clr sound bit
;************************

CLRB RB.7
;jmp

;************************
;variable delay
;************************

CLR $10
MOV $11, #2
;variable delay routine

oloop DJNZ $10, oloop
DJNZ $11, oloop

;************************
;jmp

JMP m3

flg3 JB $0E.0, m3
SETB $0E.0
;flag already set?
DJNZ $15, m3
;else set flag
;more sound bits

;************************
;jmp

D3MIN DJNZ $13, BACK
JNB $0E.1, D30SEC
;jmp

SETH $0E.2
;30 sec beep flag set? No, dcr 30 sec ctr
;
;set beep flag

;************************
;jmp

MOV $15, #2
;set for 2 seconds

m4 MOV w, RA
;

JB RA.0, flg4
;read clock port

CLRB $0E.0
;flag already set?

SETH $0E.0
;port = lo, clear flag

SETB RB.7
;set sound bit

;************************
;jmp

;************************
;variable delay
;************************

CLR $10
MOV $11, #3
;variable delay routine
tloop DJNZ $10, tloop
DJNZ $11, tloop

;************************
;jmp

CLRB RB.7
;jmp

;************************
;jmp

;************************
;variable delay
;************************

CLR $10
MOV $11, #3
;variable delay routine
tloop DJNZ $10, tloop
DJNZ $11, tloop

;************************
;jmp

JMP m4

flg4 JB $0E.0, m4
;flag already set?

SETH $0E.0
;else set flag

DJNZ $15, m4
;more sound bit

;************************
;jmp

D30SEC DJNZ $14, BACK
;jmp

BACK RETP
;jmp

;************************
;jmp

;************************
;jmp

;************************
Distance measurement System development
GP2D05 Distance Measuring Sensor – 1-bit Output

Open drain operation input

1. Distance measuring type object sensor (Distance measuring range : Optional distance can be set as threshold level by means of built-in VR)

2. Impervious to color and reflectivity of reflective object

Applications:
- Sanitary sensors (human body detection)
- OA equipment (paper detection)
- Game equipment
- For consumer products (human body detection)

3. High precision distance measurement
Distance measuring system prototype
Sharp Sensor Schematic

Block Diagram

Reflective object

IR LED

Light detector (PSD*)

LED drive circuit

Signal processing circuit

Control circuit

Reg

12kΩ

GND

VCC

VCC

Vin

Vout

VR : Threshold setting volume

* PSD : Position Sensitive Detector

Grn lead w/diode [RA0]

Yellow lead [RB7]

Black lead

Red lead
Measurement Timing

Timing Chart

Control signal ($V_{in}$) → Current [OFF] → Current [ON] → 1ms or More → Output [H] or [L]

Output ($V_o$) → MAX. 56ms
TYP. 28ms

Distance measuring operation start
Sensor Flowchart

- Need to write code for:
  - 1 mSec timer
  - 56mSec timer
  - return pulse detector routine
  - 2 Sec timer
Init & port configurations

ORG $0 ;Page 0 for jump table

DEVICE SX28L, OSC1MHZ, TURBO, STACKX, OPTIONX, IRCDIV1
IRC_CAL IRC_FAST ; calibrate internal RC clock
RESET Start
ID 'GP2D05' ;Device = SHarp GP2D05 optical sensor
FREQ 1_000_000

ORG $50
;******************************
;port direction config.
;******************************
START MODE $0F
MOV !ra, #%0000 ;all bits = out
MOV !rb, #%11111111 ;all bits = in
MOV !rc, #%00000000 ;all bits = out

;******************************
;port startup logic levels
;******************************
MOV ra, #%0001 ;all port pins = low, except RA0
MOV rb, #%00000000 ;all port pins = low
MOV rc, #%10000000 ;port 7 pin = hi - LED off

MODE $0E ;pullup resistor config.
MOV !ra, #%1110 ;all bits1-3 = normal; bit 0 = pullup
MOV !rb, #%01111111 ;all bits = no pullup except rb7
MOV !rc, #%11111111 ;all bits = normal

MODE $0D ;logic level config.
MOV !ra, #%1111 ;bit0 = TTL,
MOV !rb, #%11111111 ;all TTL
MOV !rc, #%11111111 ;all TTL
Main

; Main loop
.pulse CALL trig ; start measurement pulse
  CALL ms56 ; wait 56 msecs
  CALL detect ; look for detection signal
  CALL ms2k ; 2 sec pause
  JMP pulse ; loopback
Subroutines

;******************************
; 1 millisecond timer
;******************************
msec  MOV  $0C, #250 ; set reg to rollover from zero to start
      MOV  $0D, #2  ; set loop for 2x
Loop  DECSZ $0C ; 250 usecs each time test = 0
      JMP  Loop
      DECSZ $0D
      JMP  Loop
RET ; done

;******************************
; 56 millisecond timer
;******************************
ms56  MOV  $0E, #60 ; SET COUNTER FOR 56 TIMES
mslp  CALL  msec  ; 1 msec
      CALL  msec
      DECSZ $0E
      JMP  mslp
      RET

;******************************
; 2 second timer
;******************************
ms2k  MOV  $0E, #250 ; SET COUNTER FOR 250 TIMES
inlp  CALL  msec  ; 1 msec
      CALL  msec
      DECSZ $0E
      JMP  inlp
      RET
;******************************
;    send trigger pulse to sensor
;******************************
trig   SETB    RC.7
      CALL    msec
      CLR     RA.0      ;send trigger pulse
      RET

;******************************
; return pulse detection
;******************************
detect MOV    $0F, #20      ;set counter for 14 times
quart  MOV    W, RB        ;read all bits on port B
      JNB      RB.7, getout  ;jump out if bit is low
      CALL    msec
      DECSZ     $0F
      JMP quart
      MOV    rc, %10000000    ;Blt 7 hi = LED OFF
      SETB    RA.0            ;reset trigger bit
      RET
Overview
The eZ430-Chronos is a highly integrated, wearable wireless development system based on the CC430. It may be used as a reference platform for watch systems, a personal display for personal area networks, or as a wireless sensor node for remote data collection. Based on the CC430F6137 <1 GHz RF SoC, the eZ430-Chronos is a complete featuring a 96 segment LCD display, an integrated pressure sensor and 3-axis accelerometer for motion sensitive control. The integrated wireless interface allows the Chronos to act as a central hub for nearby wireless sensors such as pedometers and heart rate monitors. The eZ430-Chronos offers temperature and battery voltage measurement and is complete with a USB-based CC1111 wireless interface to a PC. The eZ430-Chronos watch may be disassembled to be reprogrammed with custom applications and includes an eZ430 USB programming interface.
eZ430-Chronos Development Tool

Figure 1-1. eZ430-Chronos

Figure 4-1. Removing Watch Module from Housing
eZ430-Chronos Development Tool

Figure 4-3. Chronos Watch Module Attached to eZ430-RF Debug Interface

Connector J1 on the emulator (starting on top with pin 1) (see Figure 4-10).
- UART TX (from emulator to target)
- 3.6V supply for target
- TEST/SBWTCK
- RST/SBWTDIO
- GND
- UART RX (from target to emulator)

Figure 4-10. eZ430-Chronos Debug Interface
4.2 Functional Description of the Chronos Watch

The core technology behind the eZ430-Chronos watch is the CC430F6137 microcontroller with its integrated <1GHz radio. The CC430 also controls the LCD and its temperature sensor is used for temperature measurement. The only other ICs on the watch PCB are pressure and acceleration sensors and the LCD backlight driver.

---

Figure 4-7. eZ430-Chronos Watch Block Diagram

The PCB does not include an antenna; the metal frame of the watch module that surrounds the LCD is used as antenna. The antenna is tuned for best performance when the watch is worn or held in hands.
CC-430 CPU

CC430F613x Functional Block Diagram

- Unified Clock System
- ACLK
- SMCLK
- MCLK
- Ref Voltage Reference
- I/O Ports
- Port Mapping Controller
- ADC12
- CRC16
- Sys Watchdog
- MPY32
- Flash
- RAM
- RAM 4kB
- RAM 2kB
- SYSC
- Timer0, A5
- Timer1, A3
- Power Mgmt
- LDO
- Brownout
- 5CC Registers
- 3CC Registers
- RTC_A
- USCI_A0
- I2C
- LDO_B
- 96 Segments
- Mux
- USCI_B0
- SPI, I2C
- ABS128
- Security En-Decription
- Packet Handler
- Digital RSSI
- Center Sensing
- POI / LQI / CCA
- Sub-1GHz
- Radio (GFSK)
- CPU Interface
- MODEM
- Frequency Syntetizer
- RF/ANALOG
- TX & RX
- RF_P
- RF_N
Sub-1 GHz Radio

The implemented sub-1-GHz radio module is based on the industry-leading CC1101, requiring very few external components. Figure 1 shows a high-level block diagram of the implemented radio.

![Block Diagram](image)

Figure 1. Sub-1 GHz Radio Block Diagram

The radio features a low-IF receiver. The received RF signal is amplified by a low-noise amplifier (LNA) and down-converted in quadrature to the intermediate frequency (IF). At IF, the I/Q signals are digitized. Automatic gain control (AGC), fine channel filtering, demodulation bit/packet synchronization are performed digitally.

The transmitter part is based on direct synthesis of the RF frequency. The frequency synthesizer includes a completely on-chip LC VCO and a 90 degrees phase shifter for generating the I and Q LO signals to the down-conversion mixers in receive mode.

The 26 MHz crystal oscillator generates the reference frequency for the synthesizer, as well as clocks for the ADC and the digital part.

A memory mapped register interface is used for data access, configuration and status request by the CPU. The digital baseband includes support for channel configuration, packet handling and data buffering.

For complete module descriptions, refer to the CC430 Family User's Guide, literature number SLAU259.
Development systems

Texas instruments
Tiva™ C Series TM4C123G
LaunchPad Evaluation Kit
$13.50

Features

- Tiva C Series TM4C123GH6 microcontroller
- USB Micro-AB connector
- Device mode default configuration
- Host/OTG modes supportable
- RGB user LED
- Two user switches (application/wake)
- Available I/O brought out to headers on a 0.1" grid
- On-board In-Circuit Debug Interface (ICDI)
- Switch-selectable power sources
- ICDI
- USB Device
- Reset switch
- Preloaded RGB quickstart application
- Supported by TivaWare™ for C Series software including the USB library and the peripheral driver library
- Tiva C Series TM4C123G LaunchPad BoosterPack XL interface which features stackable headers to expand the capabilities of the 40-pin Tiva C Series LaunchPad evaluation platform

Atmel 8-bit
AVR364: MEGA-1284P
$31.25

Features

- Atmel® megaAVR® ATmega1284P microcontroller
  - Target controller
- Atmel AVR® AT32UC3B1256 32-bit microcontroller
  - Board controller
  - Communication gateway
- Analog input (to ADC)
  - Temperature sensor
  - Light sensor
  - RC filter
- Digital I/O
  - Three mechanical buttons
  - Four LEDs
  - Four expansion headers
- Footprints for external memory
  - Atmel AT45DB series DataFlash® serial flash
  - Atmel AT25DF series industrial standard serial data flash
- Touch
  - One Atmel QTouch® button
Development systems

ST Micro
STM32F0DISCOVERY
$9.00

Microchip
chipKIT™ UNO32
$28.64

Features
- STM32F051r8t6 microcontroller with 64kFLASH
- 8kRAM in LQFP64 package
- On board ST-link/v2 with selection mode switch for Programming and debugging
- Board power supply through USB or external power Supply (3V and 5V)
- 4 LED’s – 3.3v power, USB comm., PC8 PC9 outputs
- 2 push buttons
- Extension header
- Additional board for prototyping and extension

Key Features
- Application development using an environment based on the original Arduino IDE, modified to support PIC32 devices while still supporting the original Arduino line. Leverages existing code examples, tutorials and resources.
- Pin-out compatibility with many existing Arduino shields
- Higher performance at a lower price-point than existing solutions
- Advanced capabilities including:
  - Integrated USB (Device/Host, OTG
  - Integrated Ethernet
  - CAN
Development systems

Freescale  
i.MX53 Quick Start  
Development Board  
$49.00

Features
• i.MX53 1GHz ARM Cortex-A8 processor
• MC34708 PMIC
• 4GB microSD card with Linux image
• 5-volt power supply with worldwide adapters
• Micro USB cable
• Quick start guide
• DVD with VMware player, getting started video, demos and other documents

Arduino Uno  
$25.00

Features
• Microcontroller ATmega328
• Operating Voltage 5V
• Input Voltage (recommended) 7-12V
• Input Voltage (limits) 6-20V
• Digital I/O Pins 14 (of which 6 provide PWM output)
• Analog Input Pins 6
• DC Current per I/O Pin 40 mA
• DC Current for 3.3V Pin 50 mA
• Flash Memory 32 KB (ATmega328) of which 0.5 KB used by bootloader
• SRAM 2 KB (ATmega328)
• EEPROM 1 KB (ATmega328)
• Clock Speed 16 MHz
The Raspberry Pi is a credit-card-sized single-board computer developed in the UK by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools. The Raspberry Pi has a Broadcom BCM2835 system on a chip (SoC),[3] which includes an ARM1176JZF-S 700 MHz processor (The firmware includes a number of "Turbo" modes so that the user can attempt overclocking, up to 1 GHz, without affecting the warranty),[4] VideoCore IV GPU,[12] and was originally shipped with 256 megabytes of RAM, later upgraded to 512 MB.[13] It does not include a built-in hard disk or solid-state drive, but uses an SD card for booting and long-term storage.[14] The Foundation's goal was to offer two versions, priced at US$25 and US$35.

Operating system Linux (Raspbian, Debian GNU/Linux, Fedora, and Arch Linux ARM)[2] RISC OS, FreeBSD, NetBSD, Plan 9

Power 2.5 W (model A), 3.5 W (model B)

CPU ARM1176JZF-S (ARMv6K) 700 MHz,[3] Raspberry Pis can dynamically increase clockspeeds, and some can temporarily reach speeds up to 1 GHz.[4]

Storage capacity SD card slot

(SD or SDHC card)

Memory 256 MB (Model A)
512 MB (Model B rev 2)[5]
256 MB (Model B rev 1)
Banana Pi M64

$74.00

• 1.2 Ghz Quad-Core ARM Cortex A53 64-Bit Processor.
• 2GB DDR3 SDRAM with 733MHz.
• 8 GB eMMC storage (16,32,64 options available).
• WiFi (AP6212) & Bluetooth onboard.
Silicon Labs Thunderboard React

BGM111 Bluetooth Smart Module
- Bluetooth 4.2 compliant
- Integrated antenna and excellent RF performance
- Configurable TX power up to +8dBm
- ARM Cortex M4 core with 32 kB RAM and 256 kB Flash
- Low Energy Consumption
- Flexible MCU peripheral interfaces

Mobile app
- Mobile app for Android and iOS
- iOS app implemented in Swift
- Android app implanted in native code
- Source code available at GitHub

Mobile app features
- Beacon notifications
- View sensor data
- Control LEDs and detect button pushes
- Stream sensor data to the cloud

Cloud streaming
- Data is stored in Firebase
- Web frontend is implemented in ReactJS
  - Using Alt.js as the flux implementation
  - D3.js for data driven charts
- All source code available

Input and Sensors
- Si1133 UV Index and Ambient Light Sensor
- Si7021 Digital relative humidity and temperature sensor
- Si7201 Hall-effect Magnetic Sensor
- Invensense MPU-6500 Six-Axis (Gyro + Accelerometer) MotionTracking device
- 2 Buttons
- 2 LEDs

$29.00
Particle Photon

- Particle PØ Wi-Fi module
  - Broadcom BCM43362 Wi-Fi chip
  - 802.11b/g/n Wi-Fi
  - STM32F205RGY6 120Mhz ARM Cortex M3
  - 1MB flash, 128KB RAM
- On-board RGB status LED (ext. drive provided)
- 18 Mixed-signal GPIO and advanced peripherals
- Open source design
- Real-time operating system (FreeRTOS)
- Soft AP setup

$19.00
• ATmega328P onboard chip in QFN package
• 16MHz clock rate, 28K FLASH available
• USB bootloader with a nice LED indicator looks just like a USBtinyISP so you can program it with AVRdude and/or the Arduino IDE (with a few simple config modifications).
• Also has headers for an FTDI port for reprogramming
• Micro-USB jack for power and/or USB uploading, you can put it in a box or tape it up and use any USB cable for when you want to reprogram.
• On-board 5.0V power regulator with 150mA output capability and ultra-low dropout. Up to 16V input, reverse-polarity protection, thermal and current-limit protection.
• Power with either USB or external output (such as a battery) - it'll automatically switch over
• On-board green power LED and red pin #13 LED
• Reset button for entering the bootloader or restarting the program.
• Works with 99% of existing Arduino sketches (anything that doesn't use more than 28K, and doesn't require pins #2 and #7)
Summary

There are many types of microcontroller development systems available.

They are usually low priced and have a lot of support documentation to promote their use by hobbyists.

Once you have an application in mind, select the appropriate microcontroller or small board development system that will best meet the application criteria.

New cloud technology to unify IoT devices:

https://www.youtube.com/watch?v=DmoeQpjPu9o