
Decoding Infrared Remote Controls Using a PIC16C5X Microcontroller

*Author: William G. Grimm
Consultant*

INTRODUCTION

For many years the consumer electronics industry has been employing infrared remote controls for the control of televisions, VCR's, and cable boxes. This same technology has recently started to appear in industrial applications to eliminate keypads.

Decoding most of the infrared signals can be easily handled by PIC16C5X microcontrollers. This application note describes how this decoding may be done.

The only mandatory hardware for decoding IR signals is an infrared receiver. The use of two types is described here. Both are modular types used often by the consumer electronics industry. The first type responds to infrared signals modulated at about 40 kHz. The second responds to non-modulated infrared pulses and has a restricted range. The hardware costs of each approach will be less than two dollars.

Three PIC16C5X application programs are described, and instructions on how they can be used to create an algorithm that can decode just about any remote control signal. Each PIC16C5X application program represents a step in mapping out a pre-existing infrared format. The final application is a fully implemented example of decoding and interpreting the infrared signals of a type of Teknika TV remote.

THE THREE LAYERS OF AN INFRARED SIGNAL

The typical infrared signal used by remote controls has three layers. The names used for these layers has not been standardized. In this application note they are called the infrared, the modulation, and the serial data.

The infrared layer is the means of transmission. Infrared is light whose wavelength is too long to see. Although you cannot see the infrared beam, it behaves the same as light, so if you cannot see the target device, you cannot control it with an infrared signal. To control around corners or through opaque materials, RF, usually UHF signals are used. Although this application note does not further mention RF, much of what is presented here can be used with an RF transmission medium.

The modulation layer refers to the fact that each burst of infrared signal is often modulated at a frequency between 32.75 kHz and 56.8 kHz. This is done to diminish the effects of ambient light. This layer, however, is optional. Some infrared formats do not modulate their outputs, sending pulses of unmodulated infrared light instead. This is done to extend the remote control's battery life and to reduce the cost of the remote control device.

The serial data layer has the information containing a command. This is typically coded in the lengths of infrared bursts or in the lengths of gaps between infrared bursts. A long gap or burst is interpreted as a '1', a short gap or burst is interpreted as a '0'.

HARDWARE DESCRIPTION

The schematic in Figure 1 shows a tool that can be made to aid development of infrared receiver code. The schematic consists of a PIC16C57 connected to one of two available infrared receivers. One receiver is for non-modulated signals, the other for modulated signals. Modulated receivers are available from Sharp and LiteOn, part numbers GP1U521Y and LT-1060 respectively. The non-modulated type is available from Quality Technologies part number QSE157QT.

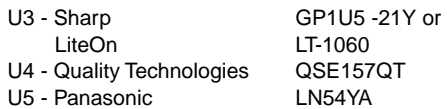
The choice of the PIC16C57 is not indicative of the processing power required for decoding. Typical IR receiver code can fit into less than half the ROM space available in a PIC16C54, and uses four RAM locations. The choice of a PIC16C57 in this case was driven by the need to store a lot of signal lengths for later reading.

A ceramic resonator clocks the PIC16C57. It will give adequate frequency accuracy to determine pulse and gap lengths. A RC network does not usually have adequate accuracy.

A button is available for resetting the PIC16C57, and four jumpers are provided to control the application start-up. The two digit display is multiplexed and driven through Q1 and Q2.

Three octal switches are used as inputs to control the OPTION register and which file is displayed.

The whole circuit derives its power from a 9V, 200 mA wall mounted supply. U1 regulates the 9V down to 5V for the PIC16C57 and associated circuitry.



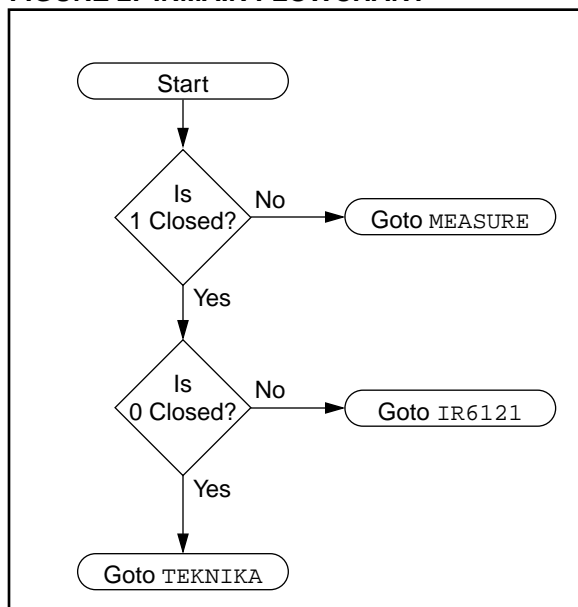
DESCRIPTION OF SOFTWARE TO AID DEVELOPMENT

This application uses four different firmware files. IRMAIN.ASM controls the selection of the three application files. The first file is MEASURE.ASM which stores the infrared burst and gap lengths into memory and allows playback of that information. IR6121.ASM decodes NEC6121 infrared format and displays the received codes on the LED display. The final file, TEKNICKA.ASM, shows the final firmware for decoding the infrared format for a Teknika Television.

IRMAIN.ASM

The firmware listed includes three applications that will aid in designing an infrared control system. IRMAIN.ASM reads jumpers 1 and 2 and directs program flow after reset to one of the three applications. Having no jumper in 2 will direct program flow to MEASURE.ASM. A jumper in 2 only will direct program flow to IR6121.ASM. Jumpers in both 1 and 2 will direct program flow to TEKNICKA.ASM. Jumpers 3 and 4 are not used.

FIGURE 2: IRMAIN FLOWCHART



MEASURE.ASM

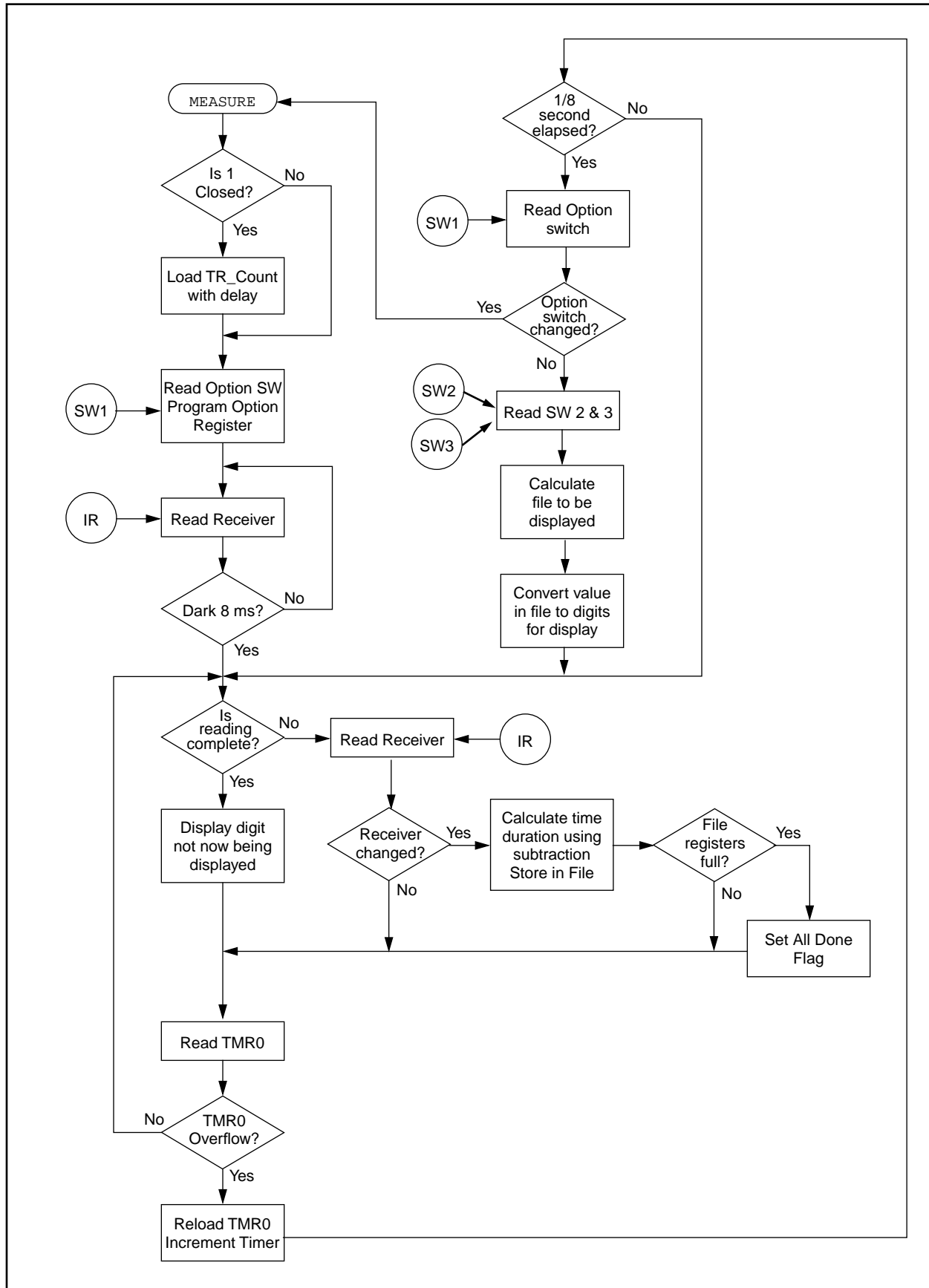
This is the most basic and most useful of the three applications. This program stores the infrared burst and gap lengths into memory, allowing playback of the measurements through the two digit display. It allows external control of the OPTION register also, through SW1. The setting of SW1 is read directly into the OPTION register prescaler value for TMR0. If SW1 is changed during program operation, the PIC16C57 resets.

Upon start-up a "hyphen" will be displayed in the left digit space until the infrared input settles to the dark logic indicating that the unit is ready to receive an infrared signal.

As an infrared signal comes in, the lengths of bursts of infrared, and the lengths of gaps between burst are stored in consecutive file locations until all four pages of the PIC16C57's memory files are filled. If a jumper had been in 1, the program throws away the first 32 pulse and gap lengths and starts storing pulse and gap lengths with the thirty third pulse length. This allows the decoding of very long formats.

When all four pages of file memory are filled with pulse and gap lengths, a number and decimal point are displayed. The decimal point indicates that the unit is done reading. The number is a gap or pulse length. SW2 and SW3 control the time sequence of the pulse or gap length displayed. These are in octal with SW3 being the more significant digit.

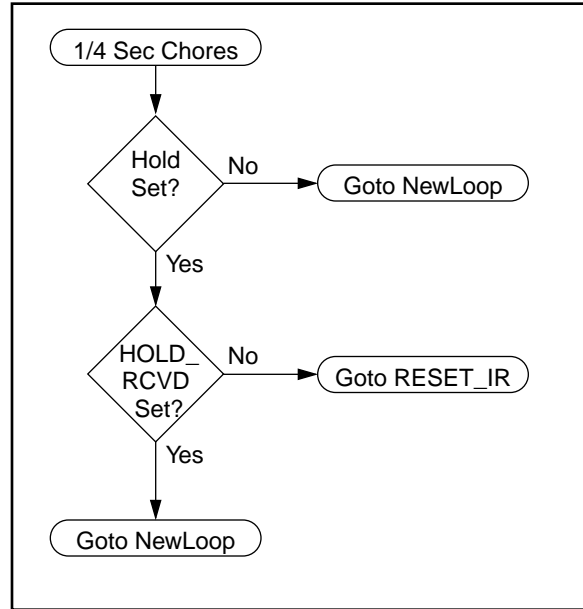
FIGURE 3: MEASURE.ASM FLOWCHART



IR6121.ASM

This is an example of the next stage in development. It uses the IR receiver, PIC16C57 clock frequency, OPTION prescaler value and characteristic time length constants that were found after using MEASURE.ASM with an infrared remote control based on the NEC6121 infrared controller[1]. The resulting algorithm is able to decode the infrared bit stream and display it as four bytes on the two digit display. The bytes are switched using SW1 (changing it will not cause this application to reset). From it, or such a program customized to your particular remote control, a list can be made of how each button on a remote control resolves to a set of bytes in memory. This allows the creation of a button lookup table.

FIGURE 4: 1/4 SECOND CHORES





TEKNIKA.ASM

This is an example of a finished product. This program fully decodes the infrared format of a Teknika Television. When a number is pressed on the remote control it is displayed on the display. When channel up or channel down is pressed, the displayed number increases or decreases.

It incorporates the final step in implementing a remote control decoder, that of cross referencing codes to button numbers. The algorithm will only respond if the first

two bytes are 14h and EBh, the characteristic of this type of Teknika television. Byte 3 and byte 4 are checked to see if they are complements. If so, byte 3 is sent through a lookup table to determine which button the received byte corresponds to, then the appropriate action is taken. The lookup table was made by using IR6121 and recording byte 3 with the button pressed. Similar tables can be made using other remote controls.

FIGURE 6: TEKNIKA.ASM FLOWCHART

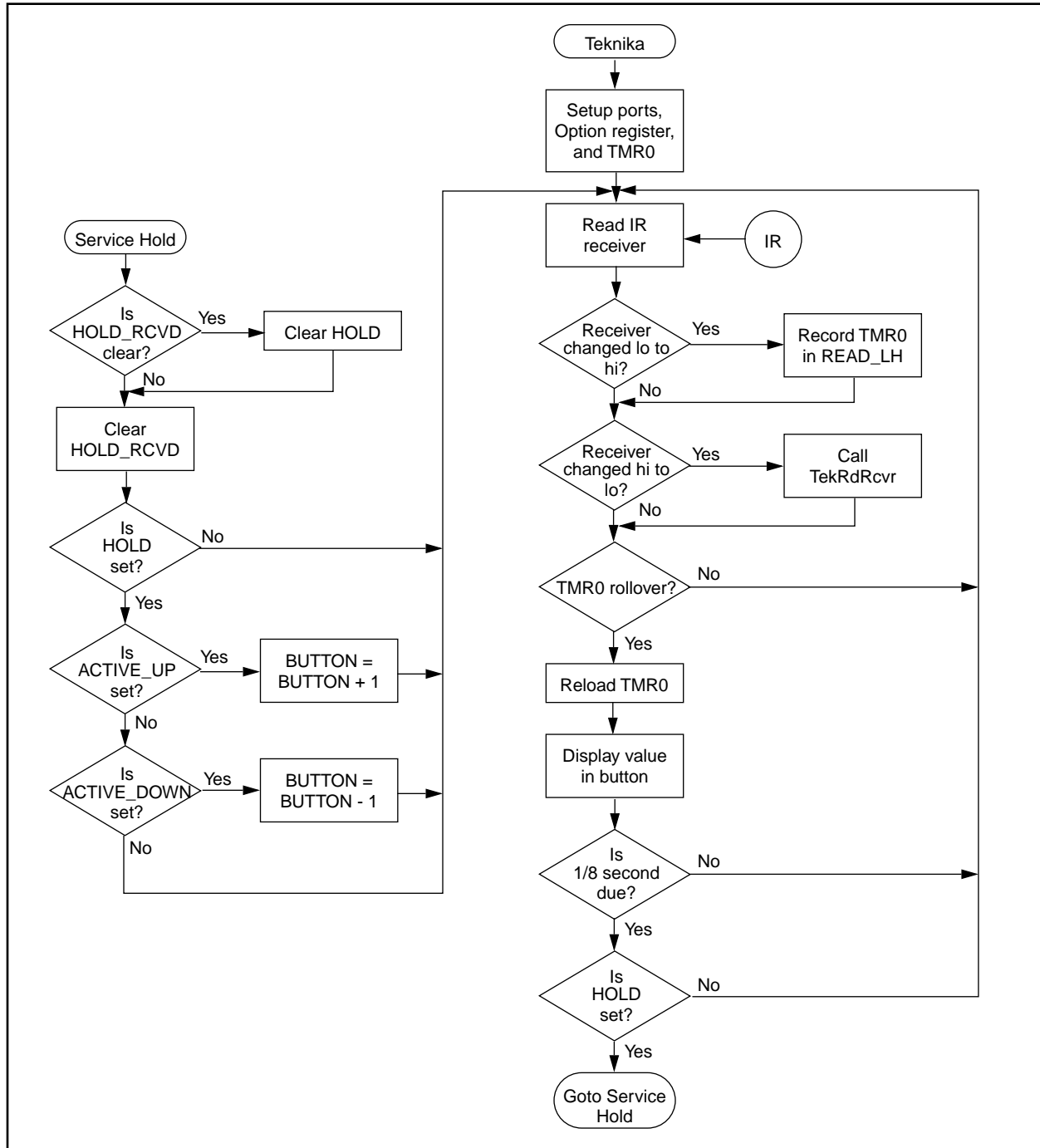


FIGURE 7: TEKRDRcvr, TEKIRReset, TEKRdAddr FLOWCHART

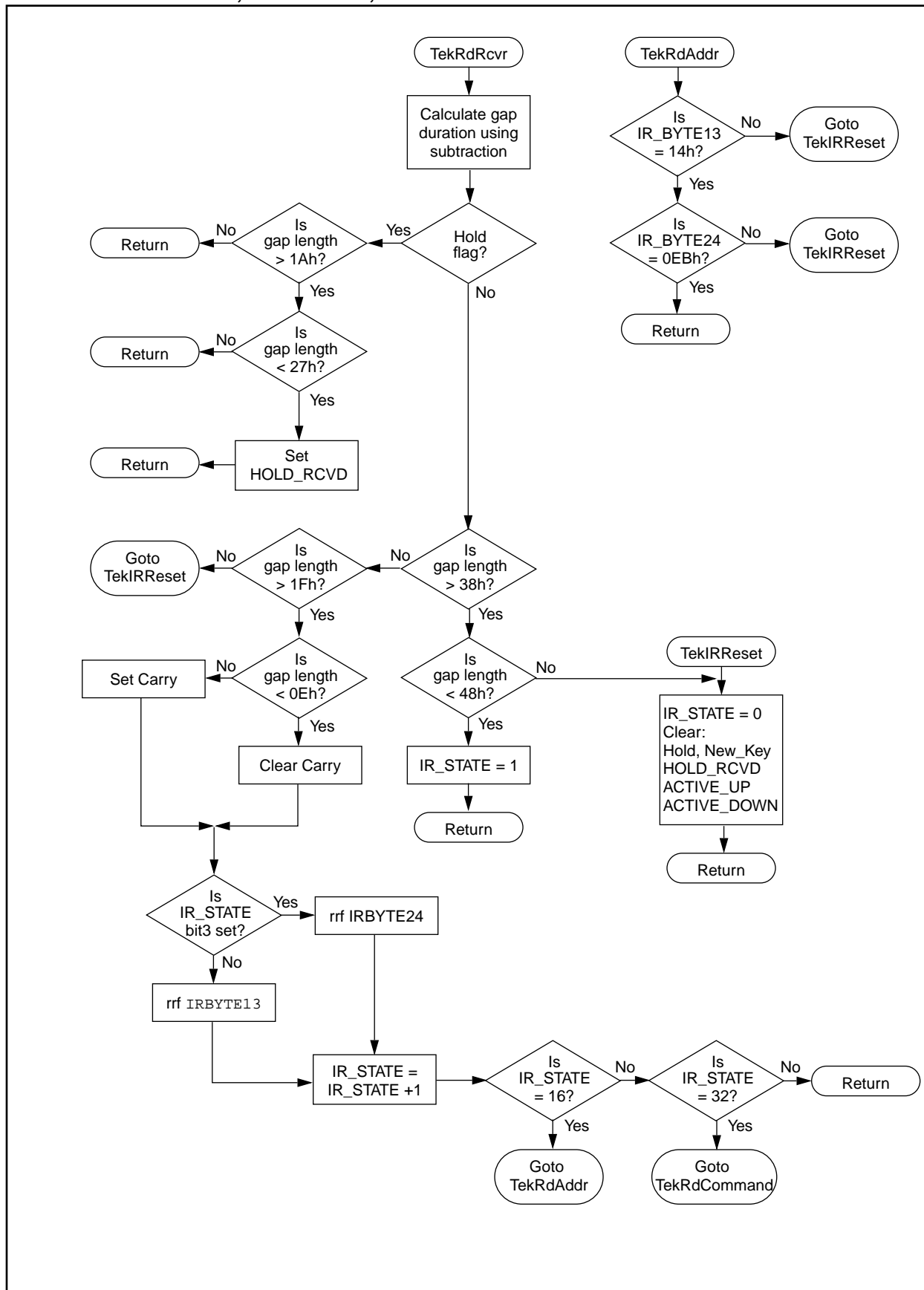
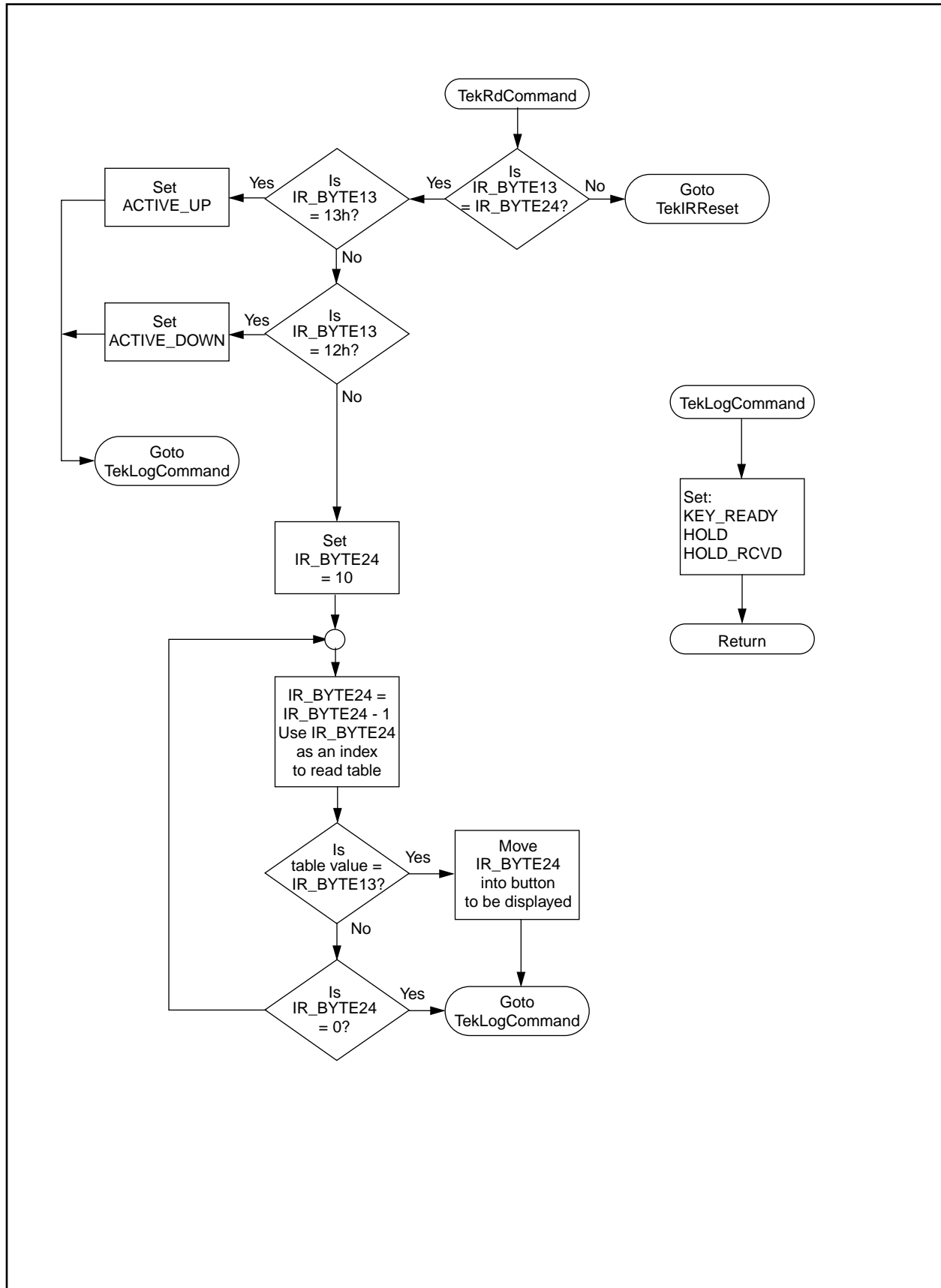


FIGURE 8: TEKCOMMAND FLOWCHART



INSTRUCTIONS ON WRITING AN ALGORITHM TO DECODE IR REMOTES

1. To design a system that uses an infrared remote control, the first step is to choose a remote control. Self designed or off the shelf, modulated or unmodulated are the primary technical decisions.
2. Once a remote control has been chosen or designed, its modulation frequency, if it has one, must be determined. This controls the kind of hardware used to receive the infrared signal.
3. The next step is to determine the time-base of the data, that is, if the pulses and gaps are short or long in reference to the PIC16C57 clock. The OPTION switch, SW1, is used to get optimum length pulse and gap counts from TMR0. This defines the value of the OPTION prescaler.
4. Fourth, definition is made as to what, in the format, defines a '1', and what, in the format, defines a '0'. This could be gap counts, pulse counts, or a combination of both.
5. Fifth, determination is made of the full length of commands. This enables the determination as to whether a button is being held down or if a new command of the same type as previous is being issued.
6. The sixth step requires the writing of code. The code will resolve the gap and pulse lengths and command lengths into bits and bytes. Each button on the remote will decode to a unique series of bits.
7. The seventh and final step takes these codes that are received and converts them to button numbers or commands, using a lookup table.

Step 1: Choosing a remote control

Depending on your application, you may choose to have an off the shelf remote control or design one yourself. Typically they have small 4-bit microcontrollers in them, preprogrammed for a serial format. Some companies such as General Instrument sell them as complete units, others such as NEC sell the main component which can be customized by external diodes to not interfere with other applications. It is also possible to program a PIC16C57 to generate a signal that can be sent to an infrared LED for transmission. Yet another approach is to use a programmable remote control to generate any number of infrared formats and use them right off the shelf to control the target device.

Step 2: Determining a modulation frequency

For this and the next step the MEASURE.ASM program will be used. To start out, use the non-modulated receiver and a PIC16C57 running the MEASURE.ASM application. Select 1 on the option selector. Press a but-

ton until the decimal point comes on. Using the jumpers switch through the memorized pulse durations that the PIC16C57 will have stored in its memory.

If all of the reading except the first are below 40h, the infrared format is a modulating one. If half or more of the values show up as 0FFh, then the remote is non-modulating.

Step 3: Determine time-base

If the remote control is modulated, switch to a demodulating IR receiver. With the option selector still at 1, press a button on the remote control again until the decimal point comes on. The series of memorized pulse durations will now probably include a lot of FFh values. If so, move the Option selector up until the values are in the 7h to 1Fh range. The Option selector has the optimum value for the option divisor to be used in the TMR0 register.

To optimize range and reliability, several demodulating receivers may be tried. These are available from Sharp or LiteOn. The modulating frequencies that are presently used are 32.75 kHz, 35.0 kHz, 36.0 kHz, 36.7 kHz, 38 kHz, 39 kHz, 40 kHz, 41.7 kHz, 48 kHz, and 56.8 kHz[2]. The most common are round 40 kHz. The best match for your remote control will give the longest range and most consistent results.

Step 4: Decoding ones and zeros

The next step is to map out the characteristic pulse and gap lengths that represent ones and zeros. By pressing the same button on the remote, write down the series of numbers read by the PIC16C57 running the MEASURE.ASM program. Each odd numbered entry is the duration of a burst of IR from the remote control. Each even numbered entry is the duration of a gap between bursts of infrared. The lengths of these gaps and bursts define ones and zeros. Their order will depend on which button is pressed. Once the characteristic lengths have been discovered for a one and a zero, an algorithm can then be created with a counter to translate the lengths into ones and zeros.

Step 5: Finding the Command Length

Press the same button again. The command duration can also be found. This is necessary to determine if a button is being held down or a new command of the same type is being issued. Most remote controls repeat the command as long as the button is held down, the repetitions separated by a long dark time, usually 0FFh on an even numbered transition. If no long even numbered counts can be found, consider that some commands can be longer than 64 transitions. The option to delay counting is available for this reason. Insert jumper 1 and MEASURE.ASM will only start storing transition times after the 32nd transition.

Step 6: Translating lengths to bits

Once the characteristic lengths of ones and zeros have been found and the length of the typical command has been found, a program can then be written to decode

these lengths to ones and zeros and display them on the two digit display. Also a HOLD flag can be created which will be true as long as the button is being held down. Usually 1/8 second between commands indicates a new command. Use this value to time out HOLD times and times between commands. IR6121.ASM is an example of a program that translates the gap lengths of the NEC6121 format to the four bytes that make up the information in each command.

Step 7: Create a button to code cross reference table

TEKNIKA.ASM implements a lookup table to translate the codes received to the actual button pressed. A counter is loaded with the highest number button that can be pressed, and the code is then looked up and compared with the code that was received. If no match, the counter is decremented until a match is found. When found, the counter then has the button number that was pressed.

Note too, that more checking may be done at this level on some formats, such as having an address, a complement of the code following the code itself for checking.

The result from all of the steps in decoding is that if a button is pressed on a remote control, that button number appears in a PIC16C57 file location. A command such as channel up or channel down will appear as two set flags, one to indicate the command, the other to indicate that it is active, HOLD. From this point the application can access these flags and files to respond appropriately.

REFERENCES

- [1] Infrared Remote Controls ICs; NEC Electronics. August 1991. Literature available 1-800-632-3531.
- [2] Sharp Optoelectronics Data Book, 1991/1992, page 961.

MPASM 01.40 Released IRMAIN.ASM 10-2-1996 10:24:32 PAGE 1

PAGE 1

[illegible]

```
00184 ; Turns off the display at the three transistors
00185 ;-----
001E   00186 DisplayOff
001E 05E6 00187      bsf     PORTB,LEFT_OFF          ; turn off the left digit
001F 05C6 00188      bsf     PORTB,RIGHT_OFF         ; turn off the right digit
0020 0800 00189      retlw    0
00190 ;
00191 ;*****
00192 ; include files
0200       00193              org             200
0200       00194 BeginMeasure
00195           include        "measure.asm"
00001               TITLE    "IR Receiver output measurement routine V0.07"
00002 ;
00003 ;*****
00004 ; File Name :            MEASURE.ASM
00005 ;*****
00006 ; Author:      William G. Grimm
00007 ; Company:    Microchip Technology
00008 ; Revision:   V0.07
00009 ; Date:       March 31, 1996
00010 ; Assembler: MPASM version 1.21
00011 ;
00012 ;*****
00013 ; Revision History:
00014 ;
00015 ;
00016 ; V0.01 Original January 9, 1995
00017 ;
00018 ; V0.02 Added overflow indication January 12, 1996
00019 ;
00020 ; V0.03 Modified to conform to Microchip specifications
00021 ; February 17, 1996
00022 ;
00023 ; V0.04 Modified for new hardware March 19, 1996
00024 ;
00025 ; V0.05 Changable option register and delay added
00026 ; March 20, 1996
00027 ;
00028 ; V0.06 Added code that creates 1/8 second time out
00029 ; for all options 1 to 7, 0 gets a 1/16 second time out.
00030 ; March 26, 1996
00031 ;
00032 ; V0.07 Improved roll over detection for long gaps and pulses
00033 ; March 27, 1996
00034 ;
00035 ; V0.08 Changed to header file March 31, 1996
00036 ;
00037 ; V0.09 Fixed bug that kept jumper 1 from being read
00038 ;
00039 ;
00040 ;*****
00041 ; Assembly options
00042           LIST          P=16C57,r=hex,N=75,C=120,T=ON
00043 ;
00044 ;*****
00045 ;
00046 ;
00047 ;
00048 ;*****
00049 ;*****
00050 ;
00051 ;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
00052           SUBTITL "RAM and Flag definitions."
00053 ;*****
00054 ; file memory location definitions
```

```

00055 ;*****
00056 ;
00057 ; full byte file memory locations
00058 ;
00000008 00059 START_COUNT EQU 08 ; TMR0 value at previous IR rcvr transition
00000009 00060 TR_COUNT EQU 09 ; transition being read
00061
0000000B 00062 TIMERM EQU 0b ; Bit5 = 1/4 sec, Bit1 = 16 millisecs.
0000000C 00063 FLAG EQU 0c ; program flags
0000000D 00064 SCALE_RECORD EQU 0d ; prescaler value is stored here
00065 ; LEFT_DIGIT EQU 0E ; defined in main routine
00066 ; RIGHT_DIGIT EQU 0F
00067 ; Files 10h-1fh,30h-3fh,50h-5fh,70h-7fh
00068 ; are used to store IR pulse and gap lengths.
00069 ;
00070 ;
00071 ;
00072 ;
00073 ;DEFINE FLAG REG FUNCTION:
00074 ; BIT # 7|6|5|4|3|2|1|0|
00075 ;-----|---|---|---|---|---|
00076 ; | | | | | | | Y -->
00077 ; | | | | | Y | --> Eighth second flag.
00078 ; | | | | Y | | --> used for math, TMR0 overdue for reload
00079 ; | | | Y | | | --> the Value START_COUNT is new
00080 ; | | Y | | | | --> measurement has overflowed
00081 ; | Y | | | | | --> Value of last IR bit received.
00082 ; Y | | | | | --> if set memory is full, stop reading
00083 ; Y | | | | | -->
00084 ;
00000001 00085 _8TH_SEC EQU 1
00000002 00086 OVERDUE EQU 2
00000003 00087 NEW_START_COUNT EQU 3
00000004 00088 OVERFLOW EQU 4
00000005 00089 LAST_IR EQU 5
00000006 00090 ALL_DONE EQU 6
00091 ;
00092 ;
00093 ;*****
00094 SUBTITL "Constant definitions."
00095 ; *****
00096 ; Definition of program constants
00097 ;
00000020 00098 SKIP_NUM EQU d'32' ; readings to skip before filing them
00000000 00099 OPTION_MASK EQU B'00000000' ; SET UP PRESCALER, WDT on 18msec.
00100 ; lowest three bits must be zero to not
00101 ; overwrite the prescaler dialed in
00102 ; externally
00103 ;
00104 ;*****
00105 SUBTITL "Timer routines."
00106 ; *****
00107 ; Timer servicing routine
00108 ; Called every 1.6 to 8 milliseconds this clears the
00109 ; watch dog, reloads TMR0
00110 ; and keeps track of relative time.
00111 ;
0200 00112 ServiceTimerM
0200 0C83 00113 movlw MSEC8 ;TMR0 = 8 milliseconds.
0201 01C1 00114 addwf TMR0,W ; Add overflow amount.
0202 0021 00115 movwf TMR0 ; /
0203 0004 00116 clrwdt
0204 076C 00117 btfss FLAG,NEW_START_COUNT ; find if measured length is too long
0205 058C 00118 bsf FLAG,OVERFLOW ; length is too long to measure
0206 046C 00119 bcf FLAG,NEW_START_COUNT ; set the flag indicating the reload
0207 02AB 00120 incf TIMERM,F ; increment the timer

```



```
0208 090F    00121      call     TimerLookup           ; get the maximum count
0209 018B    00122      xorwf     TIMERM,W             ; see if maximum count is here
020A 0743    00123      btfss     STATUS,Z               ; Is maximum count there?
020B 0800    00124      retlw     0                      ; not there return
020C 006B    00125      clrf       TIMERM              ; reset the timer
020D 052C    00126      bsf        FLAG,_8TH_SEC          ; Set the 1/8 sec flag.
020E 0800    00127      retlw     0                      ; reset and ready for 1/8 sec chores
                                00128 ;
020F                00129      TimerLookup
020F 020D    00130      movf      SCALE_RECORD,W         ; bring in the record of the option
0210 0E07    00131      andlw     7                       ; ensure lookup table is not overjumped
0211 01E2    00132      addwfc   PCL,F                    ; look up the proper timer overflow
0212 0800    00133      retlw     0                      ; this will only get 1/16 sec time out
0213 0800    00134      retlw     0                      ; option = 1
0214 0880    00135      retlw     b'10000000'            ; option = 2
0215 0840    00136      retlw     b'01000000'            ; option = 3
0216 0820    00137      retlw     b'00100000'            ; option = 4
0217 0810    00138      retlw     b'00010000'            ; option = 5
0218 0808    00139      retlw     b'00001000'            ; option = 6
0219 0804    00140      retlw     b'00000100'            ; option = 7
                                00141 ;
                                00142 ;
00143 ;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
00144                  SUBTITL "IR counter."
00145 ;
00146 ;
00147 ; *****
00148 ; IR Receiver routines
00149 ;
00150 ;-----
00151 ; ReadReceiver
00152 ; Second part of the IR receiver. It takes the present count of the
00153 ; TMR0 and subtracts the count recorded when the receiver output
00154 ; went high (START_COUNT) to find the dark pulse duration. In that duration
00155 ; will be encoded the 1, 0, HOLD, or attention.
00156 ;-----
021A                00157      ReadReceiverHi
021A 05AC    00158      bsf        FLAG, LAST_IR          ; record the IR receiver state
021B 02A9    00159      incf       TR_COUNT,F            ; 
021C 0509    00160      bsf        TR_COUNT,0            ; Times when IR rcvr is Lo are recorded
                                                ; in odd numbered locations
021D 0C3F    00162      movlw     3fh                     ; bring in highest valid address in
021E 0189    00163      xorwf     TR_COUNT,W            ; TR_count see if highest count is in
021F 0643    00164      btfscc    STATUS,Z               ; skip if not highest address
0220 05CC    00165      bsf        FLAG, ALL_DONE          ; set all done flag to stop reading
                                                ; 
0221 0A25    00167      goto       TimeIRReceiver
                                                ; 
0222                00169      ReadReceiverLo
0222 04AC    00170      bcf        FLAG, LAST_IR          ; record the IR value
0223 02A9    00171      incf       TR_COUNT,F            ; 
0224 0409    00172      bcf        TR_COUNT,0            ; Times when IR rcvr is Hi are recorded
                                                ; in even numbered locations
00173 ;
00174 ;
00175 ; Calc the length of the dark pulse,
00176 ; length of time receiver was high.
00177 ; (placed in START_COUNT)
0225                00178      TimeIRReceiver
0225 0208    00179      movf      START_COUNT,W          ; bring in the start measurement
0226 0081    00180      subwf     TMR0,W                 ; subtract the final from the start
0227 0028    00181      movwf     START_COUNT            ; gap or pulse length is now in
                                                ; START_COUNT, must be checked
0228 0C83    00183      movlw     MSEC8                   ; Base number of TMR0 count.
0229 0088    00184      subwf     START_COUNT,W          ; Subtract the base count of TMR0
022A 0603    00185      btfscc    STATUS,C               ; skip the store and toss value if neg.
022B 0028    00186      movwf     START_COUNT            ; value was positive, store
```

```

00187 ;
022C 00188 MathDone
022C 06E9 00189      btfsc    TR_COUNT,7      ; check to see if in delay
022D 0A3A 00190      goto     DoNotStore      ; do not store the value if in delay
00191 ;
00192      ; format for FSR
022E 0209 00193      movf     TR_COUNT,W      ; Setup place the count will be stored
022F 0E0F 00194      andlw    0f              ; reduce to file location
0230 0024 00195      movwf    FSR             ; place file address in FSR
0231 0584 00196      bsf      FSR,4           ; set to place in upper file group
0232 0689 00197      btfsc    TR_COUNT,4      ; set bank bit 0
0233 05A4 00198      bsf      FSR,5           ; /
0234 06A9 00199      btfsc    TR_COUNT,5      ; set bank bit 1
0235 05C4 00200      bsf      FSR,6           ; /
00201 ;
0236 0CFF 00202      movlw    0ffh            ; bring in the overflow indication
0237 078C 00203      btfss    FLAG,OVERFLOW   ; skip loading of result if overflowed
0238 0208 00204      movf     START_COUNT,W    ; bring in the measurement
0239 0020 00205      movwf    INDF            ; store it for display using
00206      ; indirect addressing
023A      00207 DoNotStore
023A 0201 00208      movf     TMRO,W           ; bring in the count now
023B 0028 00209      movwf    START_COUNT      ; store it for next time
023C 056C 00210      bsf      FLAG,NEW_START_COUNT
00211      ; set ind flag that START_COUNT is new.
00212      ; this flag is used to determine if the
00213      ; pulse has gone on too long to measure
023D 048C 00214      bcf      FLAG,OVERFLOW   ; clear any overflow indication
00215 ;
023E 0800 00216      retlw    0
00217 ;
00218 ;
00219 ;*****
00220 ; The following code segments are called by the executive
00221 ; every 1/8 second and every two seconds
00222 ;
023F      00223 EighthSecondChores          ; all that needs doing every 1/8 sec
00224      ; can be placed in this subroutine
023F 042C 00225      bcf      FLAG,_8TH_SEC   ; clear the time out flag
00226 ;
0240 0246 00227      comf     PORTB,W         ; read the dial settings
00228      ; the requested memory location is in W
0241 0024 00229      movwf    FSR
00230      ; Following formats the FSR to point to
00231      ; the selected file w/ the IR pulse
0242 04C4 00232      bcf      FSR,6           ; or gap length
0243 06A4 00233      btfsc    FSR,5           ; move bit 5 to 6
0244 05C4 00234      bsf      FSR,6           ; if 5 was 1, set 6
0245 04A4 00235      bcf      FSR,5           ;
0246 0684 00236      btfsc    FSR,4           ; move bit 4 to 5
0247 05A4 00237      bsf      FSR,5           ; if 4 was high, set bit 5 of fsr
0248 0584 00238      bsf      FSR,4           ; Format for FSR, upper bank of bytes
00239 ;
0249 04A3 00240      bcf      STATUS,PA0       ; get ready to call from page 1
024A 04C3 00241      bcf      STATUS,PA1       ; /
00242 ;
024B 0380 00243      swapf    INDF,W           ; bring in IR measurement to be disp'd
024C 0900 00244      call     LookUpDigit
024D 002E 00245      movwf    LEFT_DIGIT      ; display more significant digit
00246 ;
024E 0200 00247      movf     INDF,W           ; bring in IR measurement to be disp'd
024F 0900 00248      call     LookUpDigit
0250 002F 00249      movwf    RIGHT_DIGIT     ; display less significant digit
00250 ;
00251      if BeginMeasure==200                ; return the bits to this page
0251 05A3 00252      bsf      STATUS,PA0       ; page 1

```

```

0252 04C3 00253      bcf      STATUS,PA1      ; /
00254      endif
00255      if BeginMeasure==400
00256          bcf      STATUS,PA0      ; page 2
00257          bsf      STATUS,PA1      ; /
00258      endif
00259      if BeginMeasure==600
00260          bsf      STATUS,PA0      ; page 3
00261          bsf      STATUS,PA1      ; /
00262      endif
00263      ;
0253 0245 00264      comf      PORTA,W      ; bring in the req'd prescale value
00265      ; from the dial, reverse sense
0254 0E07 00266      andlw     7      ; AND w/ highest possible prescale value
0255 018D 00267      xorwf     SCALE_RECORD,W ; compare the prescale dial setting
00268      ; with the original one
0256 0743 00269      btfs     STATUS,Z      ; skip if the same
0257 0A61 00270      goto     StartMeasure ; restart the application if different
0258 0AAD 00271      goto     DoneEighthSecondChores
00272      ;
00273      ;
0259      00274      ClearRam      ; clears memory at reset
0259 0024 00275      movwf     FSR      ; place in fsr for indirect addressing.
025A 003F 00276      movwf     1fh      ; when zero, memory init is done.
025B      00277      MemoryInitLoop
025B 02A4 00278      incf      FSR,F      ; increment to the next memory location
00279      ; to be initialized.
025C 0060 00280      clrf      INDF      ; clear memory location.
025D 021F 00281      movf      1fh,w      ; Has top memory location zeroed yet?
025E 0743 00282      btfs     STATUS,Z      ; /
025F 0A5B 00283      goto     MemoryInitLoop ; /
0260 0800 00284      retlw     0
00285      ;
00286      ;
00287      ;*****
00288      ;      Start HERE.
00289      ;*****
0261      00290      StartMeasure
0261 006C 00291      clrf      FLAG      ; Clear out flag bank 1.
00292      ;
0262 006B 00293      clrf      TIMERM      ; restart the TIMERM at 0
00294      ;
0263 0069 00295      clrf      TR_COUNT      ; initialize memory counter
00296      ;
0264 0C0F 00297      movlw     0f      ; start zeroing at memory location 10h
0265 0959 00298      call      ClearRam      ; clear the first bank of memory
00299      ;
0266 0C2F 00300      movlw     2f      ; start zeroing at memory location 10h.
0267 0959 00301      call      ClearRam      ; clear the second bank of memory
00302      ;
0268 0C4F 00303      movlw     4f      ; start zeroing at memory location 10h
0269 0959 00304      call      ClearRam      ; clear the third bank of memory
00305      ;
026A 0C6F 00306      movlw     6f      ; start zeroing at memory location 10h
026B 0959 00307      call      ClearRam      ; clear the fourth bank of memory
00308      ;
026C 0C0F 00309      movlw     A_CONFIG      ; setup for PORTA, in loop so
00310      ; micocontroller will never forget
026D 0005 00311      tris      PORTA      ; inputs on bit 0.
026E 0C3F 00312      movlw     B_CONFIG
026F 0006 00313      tris      PORTB      ; PORTB has outputs on bits 0,6,7;
00314      ; inputs on bits 1, 2, 3, 4, and 5.
00315      ;
0270 05E6 00316      bsf      PORTB,LEFT_OFF ; turn off both digits to read jumpers
0271 05C6 00317      bsf      PORTB,RIGHT_OFF ; /
0272 0C1E 00318      movlw     C_CONFIG1      ; configuration to read from PORTC

```

```

0273 0007 00319      tris      PORTC          ; configure PORTC to read the bits
0274 0CE0 00320      movlw     -(SKIP_NUM)    ; let inputs settle, bring in skip numb
0275 0727 00321      btfs     PORTC,SW1      ; skip if jumper 1 is not installed
0276 0029 00322      movwf     TR_COUNT      ; move the skip number to file pointer
00323 ;
0277 0C00 00324      movlw     C_CONFIG2      ; bring in config to use PORTC for disp
0278 0007 00325      tris      PORTC          ; PORTC is normally all outputs
00326 ;
0279 0245 00327      comf      PORTA,W        ; bring in the requested prescale
00328 ; value from the dial, reverse sense
027A 0E07 00329      andlw     7              ; AND w/ highest possible prescale value
027B 002D 00330      movwf     SCALE_RECORD    ; record the value of the prescaler
027C 0D00 00331      iorlw     OPTION_MASK    ; Setup prescaler for TMR0, WDT on 18ms.
027D 0002 00332      option      ; /
00333 ;
00334 ;
027E 0CBF 00335      movlw     HIPHEN          ; Disp that unit waiting for dark cond's
027F 0027 00336      movwf     PORTC          ; put the Hiphen on right digit
0280 05E6 00337      bsf      PORTB,LEFT_OFF  ; turn off left digit
0281 04C6 00338      bcf      PORTB,RIGHT_OFF ; turn ON right digit
00339 ;
0282 0C83 00340      movlw     MSEC8          ;TMR0 = 8 mSEC
0283 0021 00341      movwf     TMR0          ; /
00342 ;
0284      00343 SettlingLoop
0284 0C83 00344      movlw     MSEC8          ; Check for overflow.
0285 0081 00345      subwf     TMR0,W        ; SEE IF TMR0 < MSEC8,
0286 0603 00346      btfs     STATUS,C        ; If TMR0 < MSEC8, Overflow.
0287 0A84 00347      goto     SettlingLoop    ; No overflow, no carry, loop.
0288 0900 00348      call     ServiceTimerM   ; Keep time and reload time keeper.
0289 0765 00349      btfs     PORTA,IR        ; IR receiver quiet?
028A 006B 00350      clrf     TIMERM          ; not quiet, reset timer
028B 072C 00351      btfs     FLAG,_8TH_SEC   ; Allow out of loop if quiet for 1/8sec
028C 0A84 00352      goto     SettlingLoop    ; not quiet long enough yet
028D 006C 00353      clrf     FLAG           ; re-clear all of the flags
028E 05AC 00354      bsf      FLAG,LAST_IR    ; set the flag, receiver is now hi
00355 ;
00356 ;
028F 0CBF 00357      movlw     HIPHEN          ; Display that unit is ready to receive
0290 0027 00358      movwf     PORTC          ; put the Hiphen on right digit
0291 04E6 00359      bcf      PORTB,LEFT_OFF  ; turn on left digit
0292 05C6 00360      bsf      PORTB,RIGHT_OFF ; turn OFF right digit
00361 ;
0293 093A 00362      call     DoNotStore      ; setup timer last read for first read
00363 ;
00364 ;***** Main loop Starts here. *****
0294      00365 Main
00366 ;
00367 ;
0294      00368 InnerLoop
0294 07CC 00369      btfs     FLAG,ALL_DONE    ; update display only if memory is full
0295 0A9E 00370      goto     CheckIr        ; not full, keep reading the IR rcvr
0296 04A3 00371      bcf      STATUS,PA0      ; get ready to call from page 1
0297 04C3 00372      bcf      STATUS,PA1      ; /
0298 0912 00373      call     UpdateDisplay    ; rotate power to the next display digit
00374 ;
00375 if BeginMeasure==200 ; return the bits to this page
0299 05A3 00376      bsf      STATUS,PA0      ; page 1
029A 04C3 00377      bcf      STATUS,PA1      ; /
00378 endif
00379 if BeginMeasure==400
00380      bcf      STATUS,PA0      ; page 2
00381      bsf      STATUS,PA1      ; /
00382 endif
00383 if BeginMeasure==600
00384      bsf      STATUS,PA0      ; page 3

```

```

00385      bsf      STATUS,PA1      ; /
00386  endif
029B 07C6 00387      btfss     PORTB,RIGHT_OFF      ; skip if the right digit is off
029C 04E7 00388      bcf      PORTC,DP      ; lite the decimal to show read taken
029D 0AA6 00389      goto     ReadDone      ; memory is full, done reading receiver
00390 ;
029E      00391  CheckIr
029E 0665 00392      btfsc     PORTA,IR      ; ?IR receiver not recv'g an IR burst?
029F 06AC 00393      btfsc     FLAG,LAST_IR      ; was it receiving a burst last time?
02A0 02A2 00394      incf      PCL,F      ; Not either, skip next instruction
02A1 091A 00395      call     ReadReceiverHi      ; Record TMR0 value when the lo to hi
00396      ; transition came from the receiver
00397 ;
02A2 0765 00398      btfss     PORTA,IR      ; ?IR receiver receiving an IR burst?
02A3 07AC 00399      btfss     FLAG,LAST_IR      ; was it not receiving burst last time?
02A4 02A2 00400      incf      PCL,F      ; Not either skip next instruction
02A5 0922 00401      call     ReadReceiverLo      ; read the new information
00402 ;
02A6      00403  ReadDone
00404 ;
02A6 0C83 00405      movlw     MSEC8      ; Check for overflow.
02A7 0081 00406      subwf     TMR0,W      ; SEE IF TMR0 < MSEC8,
02A8 0603 00407      btfsc     STATUS,C      ; If TMR0 < MSEC8, Overflow.
02A9 0A94 00408      goto     InnerLoop      ; No overflow, no carry, loop.
02AA 0900 00409      call     ServiceTimerM      ; Keep time and reload time keeper.
00410 ;
02AB 062C 00411      btfsc     FLAG,_8TH_SEC      ; check for 1/8 second time out
02AC 0A3F 00412      goto     EighthSecondChores
00413      ; anything that needs doing every 1/8sec
00414      ; can go in this subroutine
02AD      00415  DoneEighthSecondChores
00416 ;
02AD 0A94 00417      goto     Main
00418 ;
00419 ;
00420
0400      00196      org      400
0400      00197  BeginIr6121
00198      include     "ir6121.asm"
00001      TITLE      "IR-NEC6121 format Remote Control Detector V0.02"
00002      SUBTITL     "Comments documentation and history"
00003 ;
00004 ;*****
00005 ; File Name :      IR6121.ASM
00006 ;*****
00007 ;      Author:      William G. Grimm
00008 ;      Company:     Microchip Technology
00009 ;      Revision:    V0.02
00010 ;      Date:       February 27, 1996
00011 ;      Assembler:  MPASM version 1.21
00012 ;
00013 ;*****
00014 ;      Revision History:
00015 ;
00016 ;
00017 ;      V0.01      Original   February 27, 1996
00018 ;
00019 ;      V0.02      Converted to Ap-note format and made into a header
00020 ;                  file March 28, 1996
00021 ;*****
00000005 00022  OPTION_CODE EQU      B'00000101'      ;SET UP PRESCALER, WDT on 18msec.
00023 ;*****
00024 ;
00025 ;*****
00026 ; file memory location definitions
00027 ;*****

```

DS00657A-page 22 © 1997 Microchip Technology Inc.

```
00094 ;  
0000001A 00095 HOLD_MIN EQU 1a ; Changed from 1e to minimize intermittency.  
00000027 00096 HOLD_MAX EQU 27 ; Changed from 23 to minimize intermittency.  
00000038 00097 HEAD_MIN EQU 38  
00000048 00098 HEAD_MAX EQU 48  
0000001F 00099 ONE_MAX EQU 1f  
0000000E 00100 ZERO_MAX EQU 0e  
  
00101 ;  
00102 ;  
00103 ;  
00104 ;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  
00105 SUBTITL "Timer Routines."  
00106 ;*****  
00107 ; Timer servicing routine  
00108 ; Called every 8 milliseconds this clears the  
00109 ; watch dog, reloads the real time clock counter  
00110 ; and keeps track of relative time.  
00111 ;  
0400 00112 SvcTimer  
0400 0C83 00113 movlw MSEC8 ;TMR0 = 8 milliseconds.  
0401 01C1 00114 addwf TMR0,W ; Add overflow amount.  
0402 0021 00115 movwf TMR0 ; /  
0403 0004 00116 clrwdt  
0404 03EB 00117 incfsz TIMER,F ; Increment the timer, Skip to two sec  
00118 ; set up if it rolls over.  
0405 0A08 00119 goto CheckMatch ; Go to other possible set ups.  
0406 0559 00120 bsf FLAG3,TWO_SEC ; Set the 2 second flag.  
00121 ;  
0407 0800 00122 retlw 0 ; sync serviced.  
0408 00123 CheckMatch ; 1/8 and 1/4 sec flags are staggered  
00124 ; for more eff use of processor time.  
0408 020B 00125 movf TIMER,W ; Bring in the timer.  
0409 0E0F 00126 andlw b'00001111' ; Check lower bits.  
040A 0F02 00127 xorlw d'2' ; 1/8sec chores called when lo nibble=2.  
040B 0643 00128 btfst STATUS,Z ; Was the low nibble not 2?  
040C 0539 00129 bsf FLAG3,EIGHTH_SEC ; No! it was 2, Set the 1/8 sec flag.  
040D 020B 00130 movf TIMER,W ; Bring in the timer.  
040E 0E1F 00131 andlw b'00011111' ; Check five lower bits.  
040F 0F19 00132 xorlw 0x19 ; 1/4 second chores called every 0x19.  
0410 0643 00133 btfst STATUS,Z ; Was the low five bits not 0x19?  
0411 0519 00134 bsf FLAG3,_4TH_SEC ; No! it was 0x19, Set the 1/4sec flag.  
0412 0800 00135 retlw 0 ; matches checked, return.  
  
00136 ;  
00137 ;  
00138 ;  
00139 ;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  
00140 SUBTITL "6121 type IR remote control reader."  
00141 ; *****  
00142 ; The following reads the IR transmitter.  
00143 ; When the IR transmitter is being read,  
00144 ; This routine takes control of the clocks  
00145 ; and suspends all other functions.  
00146 ;  
00147 ; *****  
00148 ; IR Receiver routine  
00149 ;  
00150 ;  
00151 ;-----  
00152 ; ReadAddr  
00153 ; This routine places the first two bytes received in temporary  
00154 ; locations. Normally this routine would be configured to detect  
00155 ; weather or not the received command was ment for this equipment  
00156 ;-----  
0413 00157 ReadAddr  
0413 0213 00158 movf IR_BYTE13,W ; bring in the first complete byte read  
0414 003A 00159 movwf C_BYTE 1 ; store it in the contingent first byte
```

AN657

```
0415 0214 00160      movf      IR_BYTE24,W          ; bring in the second complete byte read
0416 003B 00161      movwf     C_BYTE_2             ; store it in the contingent second byte
0417 0800 00162      retlw     0                     ; command.
00163 ;
00164 ;-----
00165 ; ReadCommand
00166 ; This routine places the third and fourth bytes in memory locations
00167 ; so they can be displayed. The first two bytes are transferred from
00168 ; their trmporary locations to locations where they too can be
00169 ; displayed. Normally this routine would be configured to decode
00170 ; the appropriate action from the received number. the third and fourth
00171 ; bytes are always complements of each other in this format. Typically
00172 ; a complementary check of these two bytes is done at this point in the IR
00173 ; reception
00174 ;-----
0418      00175 ReadCommand
0418 021A 00176      movf      C_BYTE_1,W          ; bring in the first complete byte read
0419 003C 00177      movwf     BYTE_1             ; store it to be disp'd as actual first byte
041A 021B 00178      movf      C_BYTE_2,W          ; bring in the first complete byte read
041B 003D 00179      movwf     BYTE_2             ; store it to be disp'd as actual 2nd byte
041C 0213 00180      movf      IR_BYTE13,W         ; bring in the third complete byte read
041D 003E 00181      movwf     BYTE_3             ; store it to be displayed as the 3rd byte
041E 0214 00182      movf      IR_BYTE24,W         ; bring in the fourth complete byte read
041F 003F 00183      movwf     BYTE_4             ; store it to be displayed as the 4th byte
0420 0598 00184      bsf       FLAG2,KEY_READY ; Good set received
0421 0A48 00185      goto      LogHold             ; Activate the hold for the first pass.
00186 ;
00187 ;
00188 ;-----
00189 ; ReadReceiver
00190 ; Second part of the IR receier. It takes the present count of the
00191 ; RTCC and subtracts the count recorded when the receiver output
00192 ; went high (READ_LH) to find the dark pulse duration. In that duration
00193 ; will be encoded the 1, 0, HOLD, or attention.
00194 ;-----
0422      00195 ReadReceiver
0422 04B8 00196      bcf       FLAG2,LAST_IR_STATE ; Record that the IR receiver output
00197 ; is now high
00198 ; Calc the length of the dark pulse,
00199 ; length of time receiver was high.
00200 ; (placed in READ_LH)
0423 0211 00201      movf      READ_LH,W           ; bring in the start measurement
0424 0081 00202      subwf     TMR0,W              ; subtract the final from the start
0425 0031 00203      movwf     READ_LH             ; gap or pulse length is now in
00204 ; READ_LH, must be checked
0426 0C83 00205      movlw     MSEC8              ; Base number of TMR0 count.
0427 0091 00206      subwf     READ_LH,W           ; Subtract the base count of TMR0
0428 0603 00207      btfs     STATUS,C            ; skip the store and toss value if neg
0429 0031 00208      movwf     READ_LH             ; value was positive, store
00209 ;
042A      00210 Ir6121MathDone
00211 ;
042A 0678 00212      btfs     FLAG2,HOLD          ; is it now looking for holds?
042B 0A40 00213      goto      LookForHold        ; look for HOLD
00214 ;
042C 094B 00215      call     LookForAttentionGap ; look for an attention dark pulse
042D 01E2 00216      addwf     PCL,F              ; skip if a 1 was ret'd, no atten pulse
042E 0800 00217      retlw     0                     ; a 0 ret'd, ATTEN pulse found, return
00218 ;
042F 0C1F 00219      movlw     ONE_MAX             ; Test for the max length of one.
0430 0091 00220      subwf     READ_LH,W           ; If no carry gen'd, A valid 1 is found
0431 0603 00221      btfs     STATUS,C            ; No carry means the reading is below max
0432 0A5E 00222      goto      ResetIR            ; IR no good, Above maximum is invalid.
00223 ;
0433 0C0E 00224      movlw     ZERO_MAX            ; Test for the max length of Zero.
0434 0091 00225      subwf     READ_LH,W           ; If no carry gen'd, A valid 0 is found.
```



```

00226                                ; the carry now has the newly received bit
00227                                ; shift the bit into the proper location
00228 ;
0435 0772 00229      btfss      IR_STATE,3      ; Every 8 states result in dest changes
0436 0333 00230      rrf         IR_BYTE13,F      ; this bit is a part of IR byte 1 or 3
0437 0672 00231      btfsc      IR_STATE,3      ; /
0438 0334 00232      rrf         IR_BYTE24,F      ; this bit is a part of ir byte 2 or 4
00233 ;
0439 0C01 00234      movlw      1                ; Get ready to add one to the IR STATE
043A 01F2 00235      addwf       IR_STATE,F      ; inc the state setting half carry bits
043B 0723 00236      btfss      STATUS,DC       ; skip if digit carry generated
043C 0800 00237      retlw      0                ; all done reading for now
043D 07B2 00238      btfss      IR_STATE,5      ; check to determine if the 1st and 2nd
00239                                ; bytes or 3rd and 4th bytes are now ready
043E 0A13 00240      goto       ReadAddr        ; First and second byte ready.
043F 0A18 00241      goto       ReadCommand     ; Third and fourth byte ready.
00242 ;
00243 ;-----
00244 ; LookForHold
00245 ; Reads the length of the received dark pulse and determines if
00246 ; a valid HOLD pulse has been received
00247 ;-----
0440      00248 LookForHold
0440 0C1A 00249      movlw      HOLD_MIN        ; Find if between hold and one.
0441 0091 00250      subwf      READ_LH,W      ; IF no carry is gen'd, The read is between
0442 0703 00251      btfss      STATUS,C       ; HOLD and one and as such, invalid.
0443 0800 00252      retlw      0                ; Return to main routine from invalid read
0444 0C27 00253      movlw      HOLD_MAX       ; Test for the max length of HOLD.
0445 0091 00254      subwf      READ_LH,W      ; If no carry is gen'd, get a valid hold
0446 0603 00255      btfsc      STATUS,C       ;
0447 0800 00256      retlw      0
0448      00257 LogHold
0448 0578 00258      bsf         FLAG2,HOLD      ; valid HOLD received
0449 05D8 00259      bsf         FLAG2,HOLD_RCVD ; clear bit for the next hold condition
044A 0800 00260      retlw      0
00261 ;
00262 ;-----
00263 ; LookForAttentionGap
00264 ; Reads the length of the received dark pulse and determines if
00265 ; a valid attention pulse has been received
00266 ;-----
044B      00267 LookForAttentionGap            ; look for attention dark pulse
044B 0C38 00268      movlw      HEAD_MIN       ; Find if between head and one.
044C 0091 00269      subwf      READ_LH,W      ; IF no carry is gen'd, reading is between
044D 0703 00270      btfss      STATUS,C       ; HOLD and HEAD and as such, invalid.
044E 0A55 00271      goto       CheckIRState   ; continue, no attention gap.
044F 0C48 00272      movlw      HEAD_MAX       ; Test for the max length of HEAD.
0450 0091 00273      subwf      READ_LH,W      ; If no carry is gen'd, get a valid head.
0451 0603 00274      btfsc      STATUS,C       ; A carry = a too long gap and is invalid.
0452 0A55 00275      goto       CheckIRState   ; continue, no attention gap
0453 0072 00276      clrf       IR_STATE       ; Valid Attention dark pulse. This command
00277                                ; starts the state machine looking for bits
0454 0800 00278      retlw      0                ; return to main routine, ATTEN found
0455      00279 CheckIRState
0455 0CE0 00280      movlw      0e0            ; load A mask to mask all counting states
0456 0152 00281      andwf      IR_STATE,W     ; compare with present state
0457 0743 00282      btfss      STATUS,Z       ;
0458 0800 00283      retlw      0                ; not a count state, return to main routine
0459 0801 00284      retlw      01            ; counting state, look for 1's and 0's
00285 ;
00286 ;-----
00287 ; RecordRTCCatLowToHiTransition
00288 ; First part of the IR receier. It records the time when the
00289 ; output of the IR receiver went from low to high. this creates the
00290 ; starting time for timing an IR pulse.
00291 ;-----

```

```

045A      00292 RecordRTCC_atLowToHiTransition
045A 05B8 00293      bsf      FLAG2, LAST_IR_STATE ; record that IR was last in dark pulse
045B 0201 00294      movf     TMR0,W           ; bring in the clock time
045C 0031 00295      movwf    READ_LH          ; record for when it goes back low
045D 0800 00296      retlw     0
00297 ;-----
00298 ; ResetIR
00299 ; Resets the IR state machine to ready it for receiving IR messages.
00300 ;-----
045E      00301 ResetIR
045E 0478 00302      bcf      FLAG2,HOLD        ; not seen clear the hold
045F 04D8 00303      bcf      FLAG2,HOLD_RCVD    ; clear the bit for next hold condition
0460 0072 00304      clrf     IR_STATE          ; preset IR_STATE to -1
0461 0272 00305      comf     IR_STATE,F         ; /
0462 0800 00306      retlw     0
00307 ;
00308 ;*****
00309 ; The following subroutines are called by the executive
00310 ; every 1/8 second, every 1/4 second, and every two seconds
00311 ;
0463      00312 EigthSecChores                ; all that needs doing every 1/8 sec
00313 ; can be placed in this subroutine
0463 0439 00314      bcf      FLAG3,EIGHTH_SEC  ; clear the time out flag
00315 ;
0464 0245 00316      comf     PORTA,W           ; bring in the requested prescale value
00317 ; from the dial, reverse sense
0465 0E07 00318      andlw     7                ; AND w/ highest possible prescale value
0466 0643 00319      btfsc    STATUS,Z          ; if zero, display hiphens
0467 0A7A 00320      goto     DisplayHiphens    ; was zero, display hiphens
0468 002C 00321      movwf    TEMP              ; place in temporary storage
0469 00EC 00322      decf     TEMP,F            ; dec, dial settings 1 to 4 are valid
046A 064C 00323      btfsc    TEMP,2           ; if bit2 is set dial is 5 or higher
046B 0A7A 00324      goto     DisplayHiphens    ; dial is above 5, display hiphens
046C 0C1C 00325      movlw     BYTE_1
046D 01CC 00326      addwf    TEMP,W            ; add in dial setting (between 0 and 3)
046E 0024 00327      movwf    FSR              ; place in pointer register.
00328 ; NOTE! FSR bits5,6 = clear, File page1
00329 ;
046F 04A3 00330      bcf      STATUS,PA0        ; get ready to call from page 1
0470 04C3 00331      bcf      STATUS,PA1        ; /
00332 ;
0471 0380 00333      swapf    INDF,W            ; bring in IR measurement to be disp'd
0472 0900 00334      call     LookUpDigit
0473 002E 00335      movwf    LEFT_DIGIT        ; display more significant digit
00336 ;
0474 0200 00337      movf     INDF,W            ; bring in IR measurement to be disp'd
0475 0900 00338      call     LookUpDigit
0476 002F 00339      movwf    RIGHT_DIGIT       ; display less significant digit
00340 ;
00341 if BeginIr6121==200                ; return the bits to this page
00342      bsf      STATUS,PA0                ; page 1
00343      bcf      STATUS,PA1                ; /
00344 endif
00345 if BeginIr6121==400
0477 04A3 00346      bcf      STATUS,PA0        ; page 2
0478 05C3 00347      bsf      STATUS,PA1        ; /
00348 endif
00349 if BeginIr6121==600
00350      bsf      STATUS,PA0                ; page 3
00351      bsf      STATUS,PA1                ; /
00352 endif
00353 ;
0479 0800 00354      retlw     0
047A      00355 DisplayHiphens
047A 0CBF 00356      movlw     HIPHEN           ; dial not in range, display hiphens
047B 002E 00357      movwf    LEFT_DIGIT       ; / Hiphen in left digit

```

```

047C 002F 00358      movwf    RIGHT_DIGIT      ; / Hiphen in right digit
047D 0800 00359      retlw     0
00360 ;
047E          00361 QuarterSecChores           ; all that needs doing every 1/4 second
00362           ; can be placed in this subroutine
047E 0678 00363      btfsc    FLAG2,HOLD       ; Check for HOLD condition still valid
047F 06D8 00364      btfsc    FLAG2,HOLD_RCVD   ; Check to see if a hold pulse has been
00365           ; seen in the last 1/4 second
0480 02A2 00366      incf     PCL,F
0481 095E 00367      call     ResetIR           ; reset the IR state machine and get
00368           ; ready for next
0482 04D8 00369      bcf      FLAG2,HOLD_RCVD   ; Clear the hold received flag, it is
00370           ; to be set by IR controller
00371 ;
0483 0419 00372      bcf      FLAG3,_4TH_SEC    ; clear the 1/4 second time out
0484 0800 00373      retlw     0
00374 ;
0485          00375 TwoSecChores                ; things done every two seconds
0485 0C0F 00376      movlw    A_CONFIG          ; setup for PORTA, in loop so
00377           ; microcontroller will never forget
0486 0005 00378      tris     PORTA             ; inputs are on bits 0,1, and 2.
0487 0C3F 00379      movlw    B_CONFIG          ; PORTB inputs are not used,
0488 0006 00380      tris     PORTB             ; PORTB outputs control digit drives
0489 0C00 00381      movlw    C_CONFIG2        ; PORTC is all outputs
048A 0007 00382      tris     PORTC
048B 0498 00383      bcf      FLAG2,KEY_READY   ; Routine that would interpret the key
00384           ; will clear the flag that says it is
00385           ; ready
048C 0459 00386      bcf      FLAG3,TWO_SEC     ; clear the two second time out
048D 0800 00387      retlw     0
00388 ;
00389 ;
00390 ;*****
00391 ;      Start HERE.
00392 ;*****
048E          00393 StartIr6121
048E 0985 00394      call     TwoSecChores       ; re-setup ports A, B, and C
00395 ;
048F 0C05 00396      movlw    OPTION_CODE        ;SET UP PRESCALER, WDT on 18msec.
0490 0002 00397      option   ;Clock TMR0 every 64 instruc cycles.
00398 ;
0491 0078 00399      clrf     FLAG2              ; Clear out flag bank 2.
0492 0079 00400      clrf     FLAG3              ; Clear out flag bank 3.
00401 ;
0493 0CFF 00402      movlw    Off                ; Display FF at start up
0494 003C 00403      movwf    BYTE_1             ; first byte = FF
0495 003D 00404      movwf    BYTE_2             ; second = FF
0496 003E 00405      movwf    BYTE_3             ; third = FF
0497 003F 00406      movwf    BYTE_4             ; fourth byte = FF
00407 ;
0498 0C83 00408      movlw    MSEC8              ;TMR0 = 8 mSEC
0499 0021 00409      movwf    TMR0              ; /
00410 ;
049A 095E 00411      call     ResetIR           ; get the IR ready to receive
049B 05B8 00412      bsf      FLAG2,LAST_IR_STATE ; preset the IR flag for a
00413           ; RecordRTCCatLowToHiTransition
049C 04A4 00414      bcf      FSR,5             ; File page 1
049D 04C4 00415      bcf      FSR,6             ; /
00416 ;
00417 ;***** Main loop Starts here. *****
049E          00418 IRMain
00419 ;
00420 ;
049E          00421 IRInnerLoop
049E 0665 00422      btfsc    PORTA,IR           ; ?IR rcvr not receiving an IR burst?
049F 06B8 00423      btfsc    FLAG2,LAST_IR_STATE ; was it receiving a burst last time?

```

AN657

```
04A0 02A2 00424      incf      PCL,F           ; Not either
04A1 095A 00425      call      RecordRTCC_atLowToHiTransition
                                00426           ; Record the TMR0 value when the lo to
                                00427           ; hi transition came from the receiver
                                00428 ;
04A2 0765 00429      btfss     PORTA,IR        ; ?IR receiver receiving an IR burst?
04A3 07B8 00430      btfss     FLAG2,LAST_IR_STATE ; was it not rcv'g a burst last time?
04A4 02A2 00431      incf      PCL,f           ; Not either
04A5 0922 00432      call      ReadReceiver     ; read the new information
                                00433
04A6 0C83 00434      movlw     MSEC8           ; Check for overflow.
04A7 0081 00435      subwf     TMR0,W          ; SEE IF TMR0 < MSEC8,
04A8 0603 00436      btfsc     STATUS,C        ; If TMR0 < MSEC8, Overflow.
04A9 0A9E 00437      goto      IRInnerLoop     ; No overflow, no carry, loop.
04AA 0900 00438      call      SvcTimer        ; Keep time and reload time keeper.
                                00439 ;
04AB 04A3 00440      bcf       STATUS,PA0      ; get ready to call from page 1
04AC 04C3 00441      bcf       STATUS,PA1      ; /
04AD 0912 00442      call      UpdateDisplay   ; rotate power to next display digit
                                00443 ;
                                00444 if BeginIr6121==200           ; return the bits to this page
                                00445     bsf       STATUS,PA0       ; page 1
                                00446     bcf       STATUS,PA1       ; /
                                00447 endif
                                00448 if BeginIr6121==400
04AE 04A3 00449      bcf       STATUS,PA0      ; page 2
04AF 05C3 00450      bsf       STATUS,PA1      ; /
                                00451 endif
                                00452 if BeginIr6121==600
                                00453     bsf       STATUS,PA0       ; page 3
                                00454     bsf       STATUS,PA1       ; /
                                00455 endif
                                00456 ;
04B0 07C6 00457      btfss     PORTB,RIGHT_OFF ; Is display ready to display HOLD?
04B1 0778 00458      btfss     FLAG2,HOLD      ; IS the hold active?
04B2 0AB4 00459      goto      NotHold         ; do not turn on lite for HOLD indicate
04B3 04E7 00460      bcf       PORTC,DP        ; TURN on LED flag, show HOLD is active
04B4      00461 NotHold
                                00462 ;
04B4 0639 00463      btfsc     FLAG3,EIGHTH_SEC ; check for 1/8 second time out
04B5 0963 00464      call      EighthSecChores ; all that needs doing every 1/8 second
                                00465           ; can go in this subroutine
                                00466 ;
04B6 0619 00467      btfsc     FLAG3,_4TH_SEC  ; check for 1/4 second time out
04B7 097E 00468      call      QuarterSecChores ; all that needs doing every 1/4sec
                                00469           ; can go in this subroutine
                                00470 ;
04B8 0659 00471      btfsc     FLAG3,TWO_SEC   ; check for two second time out
04B9 0985 00472      call      TwoSecChores    ; all that needs doing every two secs
                                00473           ; can go in this subroutine
                                00474
04BA 0A9E 00475      goto      IRMain
                                00476 ;
0600      00199      org       600
0600      00200 BeginTeknika
                                00201      include      "teknika.asm"
                                00001          TITLE      "IR-Technica TV format Remote Control Detector V0.01"
                                00002          SUBTITL     "Comments documentation and history"
                                00003 ;
                                00004 ;*****
                                00005 ; File Name :      TEKNIKA.ASM
                                00006 ;*****
                                00007 ;      Author:      William G. Grimm
                                00008 ;      Company:     Microchip Technology
                                00009 ;      Revision:    V0.01
                                00010 ;      Date:        March 31, 1996
```

```

00011 ;      Assembler: MPASM version 1.21
00012 ;
00013 ;*****
00014 ;      Revision History:
00015 ;
00016 ;
00017 ;      V0.01      Original      March 28, 1996
00018 ;
00019 ;      V0.02      repaired bug that kept HOLD from operating
00020 ;                  March 31, 1996
00021 ;
00022 ;      V0.03      modified gpa and pulse length subtraction
00023 ;                  March 31, 1996
00024 ;
00025 ;*****
00026 ;      OPTION_CODE EQU      B'00000101'      ;SET UP PRESCALER, WDT on 18msec.
00027 ;                  Same as IR6121
00028 ;*****
00029 ;
00030 ;*****
00031 ; file memory location definitions
00032 ;*****
00033 ;
00034 ; full byte file memory locations
00035 ;      (those commented out are defined in IR6121 or IRMAIN)
00036 ;
00037 ;      TIMER      EQU      0b      ; Bit5 = 1/4 second, Bit1 = 16 millisecs.
00038 ;      BUTTON     EQU      0c      ; holds last value of last button pressed
00039 ;
00040 ;
00041 ;      RIGHT_DIGIT EQU      0f      ; defined in irmain
00042 ;      READ_LH     EQU      11      ; Low to high reading is stored here.
00043 ;      IR_STATE    EQU      12      ; Which bit is coming in.
00044 ;      IR_BYTE13   EQU      13      ; First byte for collecting inputs.
00045 ;      IR_BYTE24   EQU      14      ; Second byte for collecting inputs.
00046 ;
00047 ;      FLAG2       EQU      18      ; flag bank 2
00048 ;      FLAG3       EQU      19      ; flag bank 3
00049 ;
00050 ;DEFINE FLAG2 REG FUNCTION:
00051 ;      BIT # 7|6|5|4|3|2|1|0|
00052 ;-----|---|---|---|---|---|
00053 ;      | | | | | | Y | --> Command Ready.
00054 ;      | | | | | Y | | --> Command in process.
00055 ;      | | | | Y | | | --> Most Significant bit of time stamp.
00056 ;      | | | Y | | | | --> HOLD is active
00057 ;      | | Y | | | | | --> 4 bytes have been rcv'd successfully
00058 ;      | | Y | | | | | --> Value of last IR bit received.
00059 ;      | Y | | | | | | --> A Valid hold received < 1/4 sec ago.
00060 ;      Y | | | | | | | -->
00061 ; Y = DEFINED AS SHOWN (0/1)
00062 ; (commented definitions are defined elsewhere)
00063 ;HOLD_RCVD      EQU      6
00064 ;LAST_IR_STATE EQU      5
00065 ;KEY_READY      EQU      4
00066 ;HOLD           EQU      3
00067 ;STAMP_MSB      EQU      2
00068 ;CMD_PEND       EQU      1      ; A channel command is pending.
00069 ;CMD_RDY        EQU      0      ; A channel command is ready.
00070 ;
00071 ;DEFINE FLAG3 REG FUNCTION:
00072 ;      BIT # 7|6|5|4|3|2|1|0|
00073 ;-----|---|---|---|---|---|
00074 ;      | | | | | | Y | --> Quarter second flag.
00075 ;      | | | | | Y | | --> Eighth second flag.
00076 ;      | | | | Y | | | --> Two second flag.

```

```

0629 0599 00209      bsf      FLAG3,ACTIVE_DOWN    ; active channel down
062A 0643 00210      btfscl  STATUS,Z              ; skip again if no active channel down
062B 0A39 00211      goto     TekLogCommand
00212 ;
062C 0C0A 00213      movlw    d'10'                ; Look for ten possible buttons.
062D 0034 00214      movwf    IR_BYTE24            ; IR_BYTE24 is converted for use as
00215 ; a counter.
062E      00216      GuessLoop
062E 00F4 00217      decf      IR_BYTE24,F          ; decrement to next button to look for
062F 0254 00218      comf      IR_BYTE24,W          ; See if it rolled over to FF
0630 0643 00219      btfscl  STATUS,Z              ; If rollover, this not a valid command.
0631 0A39 00220      goto     TekLogCommand          ; not a listed button return.
0632 0214 00221      movf      IR_BYTE24,W          ; Bring in the counter, which is a
00222 ; guess as to what the button is.
0633 093B 00223      call      TekTable              ; NO! Get the code for guessed value
0634      00224      CheckGuess                      ; for dog biscuit
0634 0193 00225      xorwf     IR_BYTE13,W           ; look for a match with the guess and
00226 ; actual value which is in IR_BYTE13.
0635 0743 00227      btfscl  STATUS,Z              ; If it matches skip and stop looping.
0636 0A2E 00228      goto     GuessLoop              ; No match. Guess again.
0637      00229      TekLogButton
0637 0214 00230      movf      IR_BYTE24,W           ; bring in the count
0638 002C 00231      movwf     BUTTON                ; it has the new button number
0639      00232      TekLogCommand
0639 0598 00233      bsf      FLAG2,KEY_READY        ; Good set received
063A 0A6C 00234      goto     TekLogHold             ; Activate the hold for the first pass.
00235 ;
063B      00236      TekTable
063B 01E2 00237      addwf     PCL,F                 ; Computed jump for look-up table.
063C 0800 00238      retlw     TEK_ZERO              ; #0
063D 0801 00239      retlw     TEK_ONE               ; #1
063E 0802 00240      retlw     TEK_TWO               ; #2
063F 0803 00241      retlw     TEK_THREE            ; #3
0640 0808 00242      retlw     TEK_FOUR             ; #4
0641 0809 00243      retlw     TEK_FIVE             ; #5
0642 080A 00244      retlw     TEK_SIX              ; #6
0643 080B 00245      retlw     TEK_SEVEN            ; #7
0644 0810 00246      retlw     TEK_EIGHT            ; #8
0645 0811 00247      retlw     TEK_NINE             ; #9
00248 ;
00249 ;
00250 ;-----
00251 ; TekRdRcvr
00252 ; Second part of the IR receiver. It takes the present count of
00253 ; TMR0 and subtracts the count recorded when the receiver output
00254 ; went high (READ_LH) to find the dark pulse duration. In that duration
00255 ; will be encoded the 1, 0, HOLD, or attention.
00256 ;-----
0646      00257      TekRdRcvr
0646 04B8 00258      bcf      FLAG2,LAST_IR_STATE    ; Record that the IR rcvr output
00259 ; is now high
00260 ; Calculate length of the dark pulse,
00261 ; length of time receiver was high.
00262 ; (placed in READ_LH)
0647 0211 00263      movf      READ_LH,W            ; bring in the start measurement
0648 0081 00264      subwf     TMR0,W                ; subtract the final from the start
0649 0031 00265      movwf     READ_LH                ; gap or pulse length is now in
00266 ; READ_LH, must be checked
064A 0C83 00267      movlw     MSEC8                 ; Base number of TMR0 count.
064B 0091 00268      subwf     READ_LH,W            ; Subtract the base count of TMR0
064C 0603 00269      btfscl  STATUS,C              ; skip the store and toss value if neg
064D 0031 00270      movwf     READ_LH                ; value was positive, store
00271 ;
064E      00272      TekMathDone
00273 ;
064E 0678 00274      btfscl  FLAG2,HOLD              ; is it now looking for holds?

```



```

064F 0A64 00275      goto      TekLookHold      ; look for HOLD
                                00276 ;
0650 096F 00277      call      TekLookAtten      ; look for an attention dark pulse
0651 01E2 00278      addwf     PCL,F              ; skip if 1 was ret'd, no atten pulse
0652 0800 00279      retlw     0                  ; a 0 ret'd, ATTEN pulse found, return
                                00280 ;
0653 0C1F 00281      movlw     ONE_MAX            ; Test for the max length of one.
0654 0091 00282      subwf     READ_LH,W          ; If no carry is gen'd, get a valid one
0655 0603 00283      btfsc     STATUS,C           ; No carry means the read is below max
0656 0A82 00284      goto      TekIRReset         ; IR no good, Above maximum is invalid.
                                00285 ;
0657 0C0E 00286      movlw     ZERO_MAX           ; Test for the max length of Zero.
0658 0091 00287      subwf     READ_LH,W          ; If no carry is gen'd, get a valid 0.
                                00288 ;
                                00289 ;
                                00290 ;
0659 0772 00291      btfss     IR_STATE,3         ; Every 8 states gives dest changes
065A 0333 00292      rrf       IR_BYTE13,F        ; this bit is a part of IR byte 1 or 3
065B 0672 00293      btfsc     IR_STATE,3         ; /
065C 0334 00294      rrf       IR_BYTE24,F        ; this bit is a part of ir byte 2 or 4
                                00295 ;
065D 0C01 00296      movlw     1                  ; Get ready to add one to the IR STATE
065E 01F2 00297      addwf     IR_STATE,F          ; inc the state setting half carry bits
065F 0723 00298      btfss     STATUS,DC          ; skip if digit carry generated
0660 0800 00299      retlw     0                  ; all done reading for now
0661 07B2 00300      btfss     IR_STATE,5         ; check to determine if the 1st and 2nd
                                00301 ;
                                00302 ;
                                00303 ;
                                00304 ;
0662 0A13 00302      goto      TekRdAddr          ; First and second byte ready.
0663 0A1C 00303      goto      TekRdCommand       ; Third and fourth byte ready.
                                00304 ;
                                00305 ;-----
00306 ; LOOK_HOLD
00307 ; Reads the length of the received dark pulse and determines if
00308 ; a valid HOLD pulse has been received
00309 ;-----
0664      00310 TekLookHold
0664 0C1A 00311      movlw     HOLD_MIN            ; Find if between hold and one.
0665 0091 00312      subwf     READ_LH,W          ; IF no carry is gen'd, The read is between
0666 0703 00313      btfss     STATUS,C           ; HOLD and one and as such, invalid.
0667 0800 00314      retlw     0                  ; Ret to main routine from invalid read.
0668 0C27 00315      movlw     HOLD_MAX           ; Test for the max length of HOLD.
0669 0091 00316      subwf     READ_LH,W          ; If no carry is gen'd, get a valid hold.
066A 0603 00317      btfsc     STATUS,C           ;
066B 0800 00318      retlw     0
066C      00319 TekLogHold
066C 0578 00320      bsf       FLAG2,HOLD          ; valid HOLD received
066D 05D8 00321      bsf       FLAG2,HOLD_RCVD    ; clear bit for the next hold condition
066E 0800 00322      retlw     0
                                00323 ;
                                00324 ;-----
00325 ; LOOK_ATTEN
00326 ; Reads the length of the received dark pulse and determines if
00327 ; a valid attention pulse has been received
00328 ;-----
066F      00329 TekLookAtten      ; look for attention dark pulse
066F 0C38 00330      movlw     HEAD_MIN            ; Find if between head and one.
0670 0091 00331      subwf     READ_LH,W          ; IF no carry is gen'd, Reading between
0671 0703 00332      btfss     STATUS,C           ; HOLD and HEAD and as such, invalid.
0672 0A79 00333      goto      TekCheckState      ; continue, no attention gap.
0673 0C48 00334      movlw     HEAD_MAX           ; Test for the max length of HEAD.
0674 0091 00335      subwf     READ_LH,W          ; If no carry is gen'd, get a valid head
0675 0603 00336      btfsc     STATUS,C           ; A carry = a too long gap = invalid.
0676 0A79 00337      goto      TekCheckState      ; continue, no attention gap
0677 0072 00338      clrf      IR_STATE           ; Valid Atten dark pulse. Command starts
                                00339 ;
                                00340 ;
0678 0800 00340      retlw     0                  ; return to main routine, ATTEN found

```

```

0679      00341 TekCheckState
0679 0CE0 00342      movlw    0e0          ; load A mask to mask all count states
067A 0152 00343      andwf    IR_STATE,W    ; compare with present state
067B 0743 00344      btfss    STATUS,Z
067C 0800 00345      retlw    0          ; not a count state, ret to main routine
067D 0801 00346      retlw    01         ; counting state, look for 1's and 0's
00347 ;
00348 ;-----
00349 ; RECORD_LH
00350 ; First part of the IR receiver. It records the time when the
00351 ; output of the IR receiver went from low to high. this creates the
00352 ; starting time for timing an IR pulse.
00353 ;-----
067E      00354 TekRecordLH
067E 05B8 00355      bsf      FLAG2, LAST_IR_STATE ; record that IR was last in dark pulse
067F 0201 00356      movf     TMRO,W          ; bring in the clock time
0680 0031 00357      movwf    READ_LH        ; record for when it goes back low
0681 0800 00358      retlw    0
00359 ;-----
00360 ; TekIRReset
00361 ; Resets the IR state machine to ready it for receiving IR messages.
00362 ;-----
0682      00363 TekIRReset
0682 0478 00364      bcf      FLAG2,HOLD      ; not seen clear the hold
0683 04D8 00365      bcf      FLAG2,HOLD_RCVD ; clear bit for the next hold condition
0684 0479 00366      bcf      FLAG3,ACTIVE_UP ; clear channel up if present
0685 0499 00367      bcf      FLAG3,ACTIVE_DOWN ; clear channel down if present
0686 0072 00368      clrf     IR_STATE        ; preset IR_STATE to -1
0687 0272 00369      comf     IR_STATE,F      ; /
0688 0800 00370      retlw    0
00371 ;
00372 ;-----
00373 ; TekSvcHold
00374 ; Uses the HOLD to increment or decrement the BUTTON number.
00375 ;-----
0689      00376 TekSvcHold
0689 0679 00377      btfsc    FLAG3,ACTIVE_UP ; is Channel up now present?
068A 0A8E 00378      goto     IncButton      ; Yes, increment button
068B 0699 00379      btfsc    FLAG3,ACTIVE_DOWN ; is Channel Down now present?
068C 0A94 00380      goto     DecButton      ; Yes, Decrement button
068D 0800 00381      retlw    0          ; neither now active
068E      00382 IncButton
068E 02AC 00383      incf     BUTTON,F        ; increment button
068F 0C0A 00384      movlw    d'10'          ;
0690 008C 00385      subwf    BUTTON,W        ; Compare with 10
0691 0603 00386      btfsc    STATUS,C        ; is BUTTON < 10?
0692 006C 00387      clrf     BUTTON          ; No recycle
0693 0800 00388      retlw    0
0694      00389 DecButton
0694 00EC 00390      decf     BUTTON,F        ; Decrement button
0695 024C 00391      comf     BUTTON,W        ; Roll to FF?
0696 0743 00392      btfss    STATUS,Z        ; ship if roll over
0697 0800 00393      retlw    0
0698 0C09 00394      movlw    d'9'          ; recycle on zero
0699 002C 00395      movwf    BUTTON          ; /
069A 0800 00396      retlw    0
00397 ;
00398 ;*****
00399 ; The following subroutines are called by the executive
00400 ; every 1/8 second, every 1/4 second, and every two seconds
00401 ;
069B      00402 TekEighthSec          ; all that needs doing every 1/8 sec
00403 ; can be placed in this subroutine
069B 0439 00404      bcf      FLAG3,EIGHTH_SEC ; clear the time out flag
00405 ;
069C 0800 00406      retlw    0

```

```

00407 ;
069D 00408 TekQuarterSec ; all that needs doing every 1/4 second
00409 ; can be placed in this subroutine
069D 0678 00410 btfsc FLAG2,HOLD ; Check for HOLD condition still valid
069E 06D8 00411 btfsc FLAG2,HOLD_RCVD ; Check to see if a hold pulse has been
00412 ; seen in the last 1/4 second
069F 02A2 00413 incf PCL,F
06A0 0982 00414 call TekIRReset ; reset the IR state machine and get
00415 ; ready for next
06A1 04D8 00416 bcf FLAG2,HOLD_RCVD ; Clear the hold received flag, it is
00417 ; to be set by IR controller
06A2 0678 00418 btfsc FLAG2,HOLD ; check for active hold
06A3 0989 00419 call TekSvcHold ; service the hold function
00420 ;
06A4 0419 00421 bcf FLAG3,_4TH_SEC ; clear the 1/4 second time-out
06A5 0800 00422 retlw 0
00423 ;
06A6 00424 TekTwoSec ; things done every two seconds
06A6 0C0F 00425 movlw A_CONFIG ; setup for PORTA in loop, so
00426 ; microcontroller will never forget
06A7 0005 00427 tris PORTA ; inputs are on bits 0,1, and 2.
06A8 0C3F 00428 movlw B_CONFIG ; PORTB inputs are not used,
06A9 0006 00429 tris PORTB ; PORTB outputs control digit drives
06AA 0C00 00430 movlw C_CONFIG2
06AB 0007 00431 tris PORTC ; PORTC is all outputs
06AC 0498 00432 bcf FLAG2,KEY_READY ; Routine that would interpret the key
00433 ; will clear the flag that says it is
00434 ; ready
06AD 0459 00435 bcf FLAG3,TWO_SEC ; clear the two second time-out
06AE 0800 00436 retlw 0
00437 ;
00438 ;
00439 ;*****
00440 ; Start HERE.
00441 ;*****
06AF 00442 StartTek
06AF 09A6 00443 call TekTwoSec ; re-setup ports A and B
00444 ;
06B0 0C05 00445 movlw OPTION_CODE ;SET UP PRESCALER, WDT on 18msec.
06B1 0002 00446 option ;Clock TMR0 every 64 inst cycles.
00447 ;
06B2 0078 00448 clrf FLAG2 ; Clear out flag bank 2.
06B3 0079 00449 clrf FLAG3 ; Clear out flag bank 3.
06B4 006C 00450 clrf BUTTON ; Displays Zero on reset
00451 ;
00452 ;
06B5 0C83 00453 movlw MSEC8 ;TMR0 = 8 mSEC
06B6 0021 00454 movwf TMR0 ; /
00455 ;
06B7 0982 00456 call TekIRReset ; get the IR ready to receive
06B8 05B8 00457 bsf FLAG2,LAST_IR_STATE ; preset the IR flag for a RECORD_LH
00458 ;
06B9 04A4 00459 bcf FSR,5 ; File page 1
06BA 04C4 00460 bcf FSR,6 ; /
00461 ;
00462 ;***** Main loop Starts here. *****
06BB 00463 TekMain
00464 ;
00465 ;
06BB 00466 TekInnerLoop
06BB 0665 00467 btfsc PORTA,IR ; ?IR receiver not rcv'g an IR burst?
06BC 06B8 00468 btfsc FLAG2,LAST_IR_STATE ; was it receiving a burst last time?
06BD 02A2 00469 incf PCL,F ; Not either
06BE 097E 00470 call TekRecordLH ; Record the TMR0 value when the lo to
00471 ; hi transition came from the receiver
00472 ;

```

```

0105 05A3 00217      bsf      STATUS,PA0      ; page 1
0106 04C3 00218      bcf      STATUS,PA1      ; /
          00219      endif
          00220      if BeginMeasure==400
          00221          bcf      STATUS,PA0      ; page 2
          00222          bsf      STATUS,PA1      ; /
          00223      endif
          00224      if BeginMeasure==600
          00225          bsf      STATUS,PA0      ; page 3
          00226          bsf      STATUS,PA1      ; /
          00227      endif
0107 0A61 00228      goto      StartMeasure    ; Start the IR measurement routine
0108      00229      TekOr6121
0108 0727 00230          btfs    PORTC,SW1      ; check SW1 to determine if Tek or 6121
0109 0B0D 00231          goto      Teknika      ; jumper in, Teknika
          00232      if BeginIr6121==200
          00233          bsf      STATUS,PA0      ; page 1
          00234          bcf      STATUS,PA1      ; /
          00235      endif
          00236      if BeginIr6121==400
010A 04A3 00237          bcf      STATUS,PA0      ; page 2
010B 05C3 00238          bsf      STATUS,PA1      ; /
          00239      endif
          00240      if BeginIr6121==600
          00241          bsf      STATUS,PA0      ; page 3
          00242          bsf      STATUS,PA1      ; /
          00243      endif
010C 0A8E 00244      goto      StartIr6121      ; Start the 6121 IR format decoder
010D      00245      Teknika
          00246      if BeginTeknika==200
          00247          bsf      STATUS,PA0      ; page 1
          00248          bcf      STATUS,PA1      ; /
          00249      endif
          00250      if BeginTeknika==400
          00251          bcf      STATUS,PA0      ; page 2
          00252          bsf      STATUS,PA1      ; /
          00253      endif
          00254      if BeginTeknika==600
010D 05A3 00255          bsf      STATUS,PA0      ; page 3
010E 05C3 00256          bsf      STATUS,PA1      ; /
          00257      endif
010F 0AAF 00258      goto      StartTek        ; Start the Teknika Remote decoder
          00259      ;
          00260      ; START Vector
07FF      00261          org      0x07ff
07FF 0B00 00262          goto      StartAll      ; start vector
          00263      ;
          00264          END

```

AN657

MEMORY USAGE MAP ('X' = Used, '-' = Unused)

```
0000 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX X-----
0100 : XXXXXXXXXXXXXXXXXXXX -----
0200 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0240 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0280 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX-- -----
0400 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0440 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0480 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXX-----
0600 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0640 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0680 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
06C0 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXX----- -----
07C0 : ----- -----X
```

All other memory blocks unused.

Program Memory Words Used: 629

Program Memory Words Free: 1419

Errors : 0

Warnings : 0 reported, 0 suppressed

Messages : 0 reported, 0 suppressed

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks


The Microchip name and logo, the Microchip logo, FilterLab, KEELOQ, microID, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

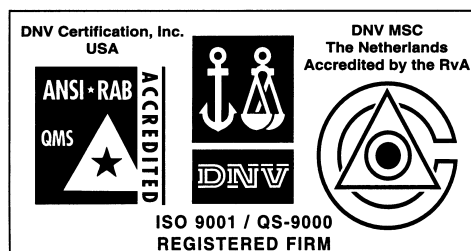
dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2002, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.



WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: <http://www.microchip.com>

Rocky Mountain

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta

500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston

2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago

333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas

4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit

Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo

2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles

18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York

150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose

Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto

6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia

Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing

Microchip Technology Consulting (Shanghai)
Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu

Microchip Technology Consulting (Shanghai)
Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou

Microchip Technology Consulting (Shanghai)
Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai

Microchip Technology Consulting (Shanghai)
Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen

Microchip Technology Consulting (Shanghai)
Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong

Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India

Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan

Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea

Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore

Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan

Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark

Microchip Technology Nordic ApS
Regus Business Centre
Lautrup høj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France

Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trappu
Batiment A - 1er Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany

Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy

Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom

Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820