A Real-Time Operating System for PIC16/17

Author: Jerry Farmer - Myriad Development Company

I. INTRODUCTION

Ever dream of having a Real-Time Kernel for the PIC16CXX family of microprocessors ? Or ever wonder what Multitasking or Threads were all about? Then this article is for you. We will explore HOW to implement all of the features of a large Real-Time Multitasking Kernel in much less space, with more control, without the large overhead of existing kernels. By planning ahead, and using the techniques outlined here, you can build your own fast, light, powerful, flexible RT kernels with just the features needed to get the job done.

Included in this article are two large examples: one on the PIC16C54, and the other on the new powerful PIC16C64. A “Remote Alarm” is implemented on the PIC16C54 as an example of a Non-Preemptive Kernel, with two asynchronous serial input sources capable of running up to 19,200 Baud along with seven sensors needing to be debounced as inputs. One more input line is monitored and causes an internal software recount. For output, this example has a LED that show eight different internal states of the “Remote Alarm”, blinking at different rates and different sequences. Last but not least, is an asynchronous Serial output capable of running at 38,400 Baud passing the inputs to the next remote alarm station. Several short and long timers are included to round out the nine cooperating tasks in this example. Please refer to Figure 2 and Appendix B.

The second example is implemented on an PIC16C64 featuring an interrupt driven semi-Preemptive kernel. This example has the serial input and output routines of the first example moved into Interrupt Service Routines (ISR) for more speed and accuracy. The interrupt capabilities of the PIC16C64 will be explored, and a Real-Time Multitasking Kernel framework will be developed. Please refer to Figure 5 and Appenidx C.

Why do I Need a Real-Time Kernel ?

Real-Time Design Techniques allow the engineer/designer to break-up large, complicated problems into smaller simpler tasks or threads. These more manageable units of code allow for faster response to important events, while prioritizing the jobs to be done in a structured well tested format. The kernel does the job of keeping the time, the peace between tasks, and keeping all the tasks’ communication flowing. More activities can be performed in the same amount of time by allowing other tasks to work while other tasks are waiting for some event to occur. Smaller code is also the result of using State-Driven techniques because much information is condensed into the state variables and code structure. If you need an example, look at the PIC16C54’s “Remote Alarm” code.

What is Multitasking Anyway ?

This is the appearance of several tasks working at the same time. Each task thinks that it owns the CPU, but this appearance is controlled by the kernel. Only one task can be running at a time, but there is undone work that can be done by other tasks not blocked. Multitasking is the orchestration of interrupts, events, communication, shared data, and timing to get a job done. Real-Time Programming is just a bunch of ideas, concepts and techniques that allow us to divide problems into units of code that are based on units of time, or events that drive a task from one state to another.

© 1994 Microchip Technology Inc.

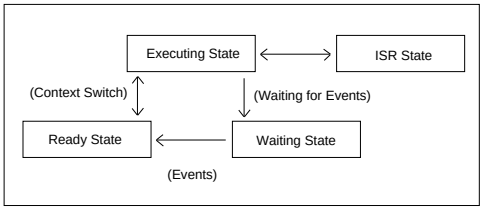
CONCEPTS

We will cover the basic concepts of kernels here so that we are using the same language when talking about this difficult topic. This article is a very quick survey on Real-Time Kernel concepts. I hope to get you thinking, reading more, and hopefully writing RT Operating Systems for your current and future projects. Many great books have been written about this very broad and interesting subject. We will refer to some of these books which have different points of view other than expressed in this paper.

Critical Section

A critical section is a shared data structure, or a shared resource, or a critical time section of code, or a non-reentrant section of code that can have only one owner that is allowed to view/change/use that section at any one time. These sections must not be interrupted during the update process. These sections must be protected so that other tasks can not get in and change the pointers/data or modify the hardware at the same time. Remember that if two tasks can get into a critical section, then data WILL be corrupted. Make sure that critical sections are small, with time for pending interrupts to get serviced. Not understanding critical sections is where the beginning RT programmers get into the most trouble. Even without interrupts, you must protect variables that are changing over time such as the byte sized variable "xmt_byte" used in the PIC16C54 example. This variable changes each time the STATE changes for the Serial Out Task. Semaphores, and Disabling Interrupts are some of the techniques used to coordinate between different tasks wanting to control a critical section. Task #4 is devoted to the proper feeding of the shared Serial Out Resource in the PIC16C54 example. Note the use of the binary semaphore "OState_B" to control Task #4, and Task #1, and variable "xmt_byte". There are several more examples of critical sections in the PIC16C64 example because of interrupts. We disable interrupts for very short time periods to protect these areas. Also in the PIC16C64 example, all critical sections are finished before checking to see if the kernel wants another task to start running instead of the current task. We will discuss in more detail how to protect critical sections later in this article.

FIGURE 1: TASK / PROCESS STATE TRANSITION DIAGRAM



Shared Resources

Data structures, displays, I/O hardware, non-reentrant routines are a few resource examples. If two or more tasks use these resources, then they are called Shared Resources and you must protect them from being corrupted. They must have only one owner, a way of telling others to wait, and maybe a waiting list for future users of that resource. A rare example of a shared resource is when there exists a critical timing sequence of input and output operations to control some hardware. You must disable interrupts before starting this sequence, and enable them upon finishing. Note that task #1 in the PIC16C64 example is an example of an "non-reentrant" routine that must be finished by the current owner before another task can use it.

Context Switch/Task Switch

When one task takes over from another, the current values of the CPU registers for this running task are saved and the old saved registers for the new task are restored. The new task continues where it left off. This is all done by the Context Switch part of the Real-Time Kernel. Each task usually has a "context switch storage area". Each task's SP (Stack Pointer pointing into its own stack) is stored there along with all the other important saved registers. The "Remote Alarm" example does not need to use a context switch because all the important registers are properly freed-up before each task is finished. The PIC16C64 example uses a similar concept, thus keeping the number of saved registers per task way down. We use an old concept called "where I came from". The variable "FROM" is used to direct the dispatcher to start up the task where it left off. This is because you can not manipulate the stack in the PIC16CXX family. This same reason is why we have a "semi- Preemptive" kernel on the PIC16C64 as an example. By the way, the faster the Context Switch is done, the better.

Scheduler

The scheduler is that part of the kernel that decides which task to run next. We will talk about several common types here in this section. This is where a lot of thinking should be done before starting your new project. By understanding the different kinds of schedulers and what features and problems each type has, you can match your problem to a creatively styled scheduler that meets your needs. For example, the PIC16C54 example shows the recalling of tasks #1-3 just before a long sequence of code is executed. More creative ways can also be done, but be careful to allow all tasks to execute in a timely fashion.

Please see Figure 1. Each task must be in "Ready State" or the "Executing State" to be considered by the scheduler to get temporary control of the CPU next.

© 1994 Microchip Technology Inc.

NON-PREEMPTIVE KERNEL

The Non-Preemptive Kernel is also called a “Cooperative Kernel” because the tasks only give-up control when they want/need to in coordination with other tasks, and events. The “Remote Alarm” example uses a Non-Preemptive Kernel type, showing that despite its reputation as being a simple kernel type, a lot can be done with it. The Non-Preemptive Kernel type is well suited for the non-interrupt type PIC16CXXs. The heart beat of the PIC16C54 example is the internal RTCC counter crossing over from high value to low value of the counter. Use the pre-scaler to adjust the time units. The very fast tasks, continually read the RTCC directly, comparing the delta of time to see if it should fire.

PREEMPTIVE KERNEL

In a Preemptive Kernel, a running task can be swapped out for a higher priority task when it becomes ready. The Preemptive Kernel relies much more on interrupts as its driving force. The context switch is at the heart of this type of kernel. To implement a true Preemptive Kernel, you must be able to manipulate the stack. This is why we implemented a “semi-Preemptive” kernel on the PIC16C64, with some of the best features of both types of kernels. We moved some of the tasks in the PIC16C54 example into ISRs to handle the I/Os. This works very well as the ISRs are very short and do most of the real work in this example. The timer0 interrupt is the heart beat for the PIC16C64 example. You must have a clock interrupt in order to make a true Preemptive kernel.

ROUND ROBIN SCHEDULER

When the scheduler finds tasks on the ready queue that have the same priorities, the scheduler often uses a technique called Round Robin scheduling to make sure each task gets its day in the sun. This means more housekeeping to get it right. This is part of the creative ways you can tailor the scheduler to fit your needs. In the PIC16C54 example, all tasks will get to run shortly after their appointed time. This means that no task will dominate all others in this simple approach. In the “olden” days of the first Real-Time Operating Systems the term was used to mean the same as “time slicing”. The Preemptive Kernels of today are a major step forward, with their priority schemes, and intertask communication capabilities.

PREEMPTIVE VS. NON-PREEMPTIVE

The Preemptive Kernel is the harder to develop, but is easier to use, and is sometimes used incorrectly. You must spend more upfront time with the Non-Preemptive Kernel but it is better for more cramped microprocessors. You get much better response time between a cause/event and the response/action for that event with a Non-Preemptive Kernel. The Preemptive Kernel is more predictable in the response times, and can be calculated as to the maximum time to complete a given job. Often the Preemptive Kernel is more expensive.

REENTRANCY

In a Preemptive Kernel, two or more tasks may want to use the same subroutine. The problem is that you can not control when a task is swapped out and when another takes its place. Thus, if a subroutine uses only local or passed variables that are stored only in each tasks’ stack, then it is call reentrant or a pure routine. No global variables or hardware may be used in such a pure routine. A way around this reentrancy requirement is to treat the whole subroutine as a critical section. Appendix D is an example of reentrant code segment as might have been used in the PIC16C54 code example.

Task Priority

Some tasks are not created equal. Some jobs must be done on time or data will be lost. Make the tasks that must get done, the highest priority, and go down the scale from there. Some kernels make you have a different priority for each task. This is a good idea and requires some thought before coding to make the design come out.

STATIC VS. DYNAMIC PRIORITIES AND PRIORITY INVERSIONS

For most embedded Real-Time Kernels, both static priorities and static tasks are used. Dynamic priorities are sometimes used to solve deadlock and other complex situations that arises from not understanding the problem and not understanding Real-Time Techniques. You should relook at how you divided the problem, and divide less so as to include the resources in question under one semaphore. You could divide the problem more to have more tasks not needing two or more resources to complete its job, and have the new tasks talk more together.

© 1994 Microchip Technology Inc.

As for Dynamic tasks, you should define the problem so as to know ahead of coding, the continuous use of all task. You will need more upfront time in the planning stage to get all tasks talking, but it is well worth it to keep Dynamic Priorities and Dynamic Tasking out of the kernel design.

Priority Inversions is a trick used to get passed a poorly designed system by allowing the inverting of the priorities to allow lower tasks to run that were blocked before. This is a cheap trick, and is best kept out of a Real-Time Kernel. Use the same techniques as outlined in this section to solve this kind of problem.

Semaphores

There are basically two types: binary and counting semaphores. The binary semaphore is used to allow only one owner, and all other tasks wanting access are made to wait. The counting semaphore, keeps a list of users that want to get access. Semaphores can be used in many ways, we will illustrate most of them in the following paragraphs. Note that you can implement counting semaphores using binary semaphores.

MUTUAL EXCLUSION

We have discussed Mutual Exclusion before, that there must be a way to exclude other tasks from gaining access to critical sections. Mutual Exclusion is the process of excluding others from access to the shared resources. To make a semaphore is a very complicated process. The semaphore's construction must be atomic. That means that once the process has started, it can not be interrupted until it has saved the name of the new owner. From there on, it knows that no one else can break-in and change owners. We have implemented the binary semaphore using bits and kernel functions to mutually exclude access in the PIC16C54 example.

In the PIC16C64 example, we also disable interrupts to get the same effect. There are at least two good ways of implementing a binary semaphore. The first and oldest way was discovered by a Dutch mathematician named Dekker. We will refer you to a book that talks more about this algorithm. The second method of implementing a binary semaphore was also discovered by another Dutchman named Dijkstra. This method uses the "testandset" type instruction and is much more important to us. We used the "dec & jump if not zero" instruction (see PIC16C64 example).

DEADLOCK

Deadlock is a condition where two or more tasks own resources that other tasks need to complete there assignment but will not release their own resources until the other tasks release theirs. Talk about cooperation. Please read section, "Static vs. Dynamic Priorities and Priority Inversions" for a discussion about such problems and ways to solve them. The root of such problems is not understanding the original problem.

SYNCHRONIZATION

Semaphores can be used to synchronize tasks so that messages can be passed between them. Also tasks can be started up by semaphores, stopped by semaphores, or started together. They are the foundation blocks for Real-Time Programming. Once you have built a binary semaphore for your kernel, you can build very complex semaphores to synchronize anything. In the PIC16C54 example, data from several sources are passed out the Serial Port Resource. Task #4 synchronizes the other tasks trying to send data out and synchronizes with task #1 to get it done. When task #1 is running, then task #4 can not run until task #1 is ready for more data to send out.

INTERTASK COMMUNICATION

We have touched on this topic already, but for large kernels, one can include more complex communication methods to pass data/messages between tasks. Much of the handshaking is done for you inside the kernel. This takes a lot more space and execution speed to implement them in a kernel.

Event Flags

We implemented Event Flags as simple bits having two states (on and off). More info can be stored per Event Flag such as time it was recorded, by who, and who the event belongs to, and whether data was lost.

Message Mailboxes

This is a nice feature to have if you have the ram space. Mailboxes allow the designer to pass messages between tasks, and allows messages to be looked at when the task is ready, and to reply telling the sender that the message was received. One message can be sent to many tasks at the same time.

Message Queues

This again is a very nice feature if you have the execution time, and the ram to implement them. This feature is related to Mailboxes, in that you can store several messages even after reading, to be processed later. If you want to only operate on the highest prioritized messages before handling the rest, this is allowed. You can be very fancy with the Mailboxes and Queues. If you have them, use them.

© 1994 Microchip Technology Inc.

Interrupts

Interrupts are one of the best inventions to come along for solving Real-Time problems. You can get very quick response to the need, and then go back to what you were doing. The only problem is that they can and do happen at the worst times. That means that you must learn how to turn them on and off to protect your critical sections. Note that before an interrupt can be handled, you must save all important registers so that you can restore them so that the kernel can restart the task where it left off. This is much like the context switch issue, but for interrupts, you must always save and restore. In the PIC16C64 example, the Status, W, and FSR registers are saved in ram because of the interrupt. The PC register is saved onto the stack by hardware.

INTERRUPT LATENCY, RESPONSE & RECOVERY

Interrupt Latency is defined as the largest time period that interrupts are disabled, plus the time it takes for the ISR to start to execute.

The Interrupt Response Time is defined for a Non-Preemptive system as Interrupt Latency plus the "context saving time". For a Preemptive system, add the execution time for the kernel to record the interrupt.

Interrupt Recovery Time for a Non-Preemptive system is defined as the time to restore the saved context and for the restarting of the task that was interrupted. Interrupt Recovery Time for a Preemptive system is the same as for the Non-Preemptive system plus the time the kernel takes in the scheduler deciding which task to run next. These measurements are how most kernels are compared with each other. The PIC16C64 example does very well in these measurements. That is because of the PIC16CXX processor and that they are mostly a Non-Preemptive system. You must keep the time you disable interrupts to a minimum in any kernel you write or any task that you write. You should break-up long sequences of instructions to allow for interrupts that are already waiting to execute.

ISR PROCESSING TIME

ISR (Interrupt Service Routine) Processing Time is defined as the time a ISR keeps control of the CPU. This amount of time should be short, and if alot of processing needs to be done in a ISR, then break up the ISR. The new ISR should now just store the new data and return. Next, create a new task and move the extra code from the old ISR into the new task. Remember that the longer you are in one interrupt, the longer you can not answer another pressing interrupt.

Nesting interrupts are where the interrupt with a higher priority can interrupt a lower priority interrupt. Care must be used, as different interrupts may have critical sections too, and disabling interrupts must be used here too to protect critical sections. Nesting of interrupts may not exist on all microprocessors, such as the PIC16CXX family.

NON-MASKABLE INTERRUPTS

On some microprocessors, you can enable/disable selected interrupts, such as on the PIC16/17 family. This is a great tool to control the flow of data into the system and out. Some systems have what is called Non-Maskable Interrupts. Here you can not turn them off by software masking. These NMIs as they are call for short, are used as clock Ticks, because you do not want problems with complex critical sections on a interrupt that you can not turn off. The PIC16CXX family does not have any NMIs. NMIs are not as useful as maskable interrupts.

CLOCK TICK

The Clock Tick, is the heart beat of the system. This is how the kernel keeps time (relative & absolute). This is how the kernel is restarted to see if there is a delay that has finished, so that that task can be moved into the ready state. In the PIC16C54 example, the RTCC clock is used. In the PIC16C64 example, the timer0 is used. You must have a clock interrupt in order to make a true Preemptive kernel. This is the other reason why we implemented a Non-Preemptive Kernel on the PIC16C54 - no clock interrupt.

© 1994 Microchip Technology Inc.

ANALYSIS OF CODE EXAMPLES

These sections are the real meat of this article. In these sections we will explain how the concepts are put to practical use line by line in each of the two main examples - PIC16C54 (Appendix C) and PIC16C64 (Appendix D).

We will also examine a short reentrant code example in Appendix B. We will give some ideas on how to expand the examples and how far and how fast the examples can be pushed. Be sure to read both sections on the two examples.

The “Remote Alarm” application has many interesting features. The concept is to have as many tiers of units like a tree feeding into the lower level units the status of each of the larger branches to one central point. Each unit can detect any changes in status before the intruder shuts that unit down, or tampers with it. If any unit’s power or wires connecting it down the tree are cut, the lack of the flow of status and passwords would be noticed in five seconds and reported down the line. The two Serial Input lines per unit receive the status and passwords from it’s two larger branches, checking the data and passing the info down the line by its own Serial Output line. The seven input status lines are debounced in these examples, showing the technique.

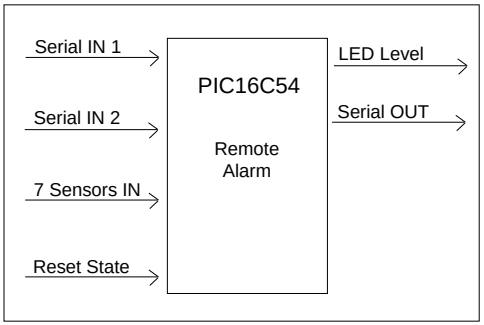
The LED on each unit reports the status at that node as to the importance of its own seven input status lines and the status flowing down the line. The level indication outputted on the LED continues at the last highest level until either a reset is received on the “Reset State” line or five minutes of no new activity on the seven input status lines are received. When either of these two events occur, the level of the LED output is adjusted to the current level of input. Some of the features are changed for this article. See Figures 2 and 5.

Another Embedded System use of this type of “Remote Alarm” application is that of placing the unit on the outside of a safe. Hopefully the intruder would be detected before arriving at the unit itself. The continuous stream of status and passwords to the larger unit inside would slow down any simple theft.

PIC16C54 - “Remote Alarm” Example

This example is a cross between a true application and an attempt to show new concepts and some extra features for show. Some of the application specific code has been removed to show more clearly the possibilities of a Real_Time Operating System on the PIC16/17 family. We chose the Baud rate for the Serial output to be twice the speed of the two Serial inputs because it is harder to accurately output a precise Serial Output than it is to monitor Serial inputs.

FIGURE 2: REMOTE ALARM - PIC16C54 EXAMPLE



This example operates at 4Mhz. By simply increasing the crystal speed to 8MHz, the two Asynchronous input Serial Baud rates increase from 4800 Baud to 9600 Baud. The Serial Output Baud rate increases from 9600 Baud to 19,200 Baud. By increasing the crystal speed to 16MHz, it will increase the Baud rates to 19,200 Baud for the two independent Asynchronous inputs, and increase the baud rate for the Asynchronous Serial output to 38,400 Baud. By adjusting the constants in the code for the Serial routines, other Baud rates can be achieved at other crystal speeds. Note that you must use a very stable crystal setup and NOT an RC combination to run these examples.

We will now give a quick outline of the PIC16C54 code example. Lines 1-85 are the equates for this program. Lines 88-95 are simple jump tables so as to save some of the precious “first 256 bytes” of each page. The Serial Output Routines - Task #1 are in lines 97-159. Task #7’s subroutines start at line 160 and continue to line 277. In this section, the LED output is controlled. The subroutine QCheck_T123, lines 278-301, is used to allow the checking of Tasks #1-3 to see if they are ready to execute before a long section of code in a slower Task is about to be executed. This is a creative way for the Kernel’s Scheduler to makes sure that the highest Prioritized Tasks get serviced before the less important tasks get executed. Task #2 starts at line 302. This task reads the Serial Input #1 for Asynchronous data. Task #2 can be described as a State Machine for outputting a byte Serially. Task #3 interrupts the code of Task #2 at line 333 and continues until line 362. Task #3 also reads the Serial Input but on input #2. Task #2’s subroutines continue at line 363 and continue until line 423. Task #3’s subroutines continue at line 424 and continue until line 484 is reached. The main or starting code is started at line 485. From that line to line 515, all variables are initialized, and all tasks are initialized at this time also.

© 1994 Microchip Technology Inc.

The Main Loop is started at line 516 and ends at line 665. This is where the real action is done. Each task checks the time to see if theconditions are correct for it to run. The tasks that are not Blocked, andhave a job to do now are in a Ready State. In the Main Loop, we check the current state of each task in order of Priority (1-9). If ready, we do a very simple Task Switch and place that task in the Executing State/Running State. Several time unit changes take place in the Main Loop. Tasks #1-4 use 2 uSecs as a time base by reading the RTCC directly. A time unit change takes place at lines 562-575 to 512 uSecs per unit for Tasks #5-6. Another time unit change takes place for Tasks #7-9, to 131072 uSecs per unit. For Tasks #5-9, each task counts the time units and compares them to their standard for activation or activity. Task #4 starts at line 538 and finishes at line 561. Task #4 controls the feeding of Task #1 from several other tasks that want data to be outputted. It uses several Semaphores to make sure that Task #1 is not bothered until it is ready for another byte. Task #5 monitors the Level Reset Line, and is always running. It simply resets the status of the LED, to be recalculated in Task #6. Task #5 runs through lines 576-581, and is very short. Lines 582-611 represent Task #6. Here we debounce the seven sensor input lines, leaving the current standard in the variable "Old_RB". Task #6 asks/Signals Task #4 to output the current standard out the Serial pin. Task #7's main code is lines 621-628. Task #8 is a five second lack of activity timer, and exists in lines 629-645. If no data comes from either of the two input Serial lines, then Task #8 Signals Task #4 to send a special byte to be outputted by Task #1. This Signals the next "Remote Alarm" of the lack of communication between units. The last task is Task #9. This is a five minute lack of Severe Errors the from Sensor Reset Timer. Lines 646-663 compose Task #9. Subroutine Do_D_H_E_L starts at line 667 and continues through to line 692. This routine determines the Highest Error Level, and passes Task #7, the current state, to output on the LED. Lines 693-703, clear the registers #7-1Fh. The "jump at Power-On" code is the last lines 705-706.

The following sections describe in more detail how and what each part of the code does and why. The code segment lines 1-87 are explained in this paragraph. Line 4 tells the MPASM assembler which PIC16/17 you are using. The include file "PICREG.H" follows with the equates and assignments to make the code more readable and changeable. You should use equates that relate symbols to each other. The Constants - lines 10-12 are the values to change for different Baud rates. They represent the Bit Times for the Baud rates divided by 2 minus some latency factor. You might have to adjust the "Fudge Factor" and other values to fine tune the performance. The value used for the "Fudge Factor" is related to the longest path of code. Lines 21-24 are an experiment that allows a simple name to be associated to a single bit. This allows for easily changeable assignments. Lines 30-54 are the variable assignments. Variables (lines 35-39) are used as time counters. They count the number of units of time, and are compared to literals to see if an Event has just happened. The bits defined in lines 57-64 are used as Binary Semaphores. They keep Critical Sections of data protected. We will see them in action later in the code. The bits defined in lines 67-73 are error flags. They define the current or last error states of the Serial routines, and whether data was lost coming in or out. The section of equates in lines 76-85 are used to define the different LED activity. They are used by Task #7 to keep the LED blinking. In lines 89-94, we try to save the all important first 256 bytes of any page.

Task #1 outputs a byte Asynchronously over the Serial Output pin. Task #1 is started at line 98. The time units used for Tasks #1-4 are 2μS. We first sample the RTCC and store the count. When Tasks #1-4 are then allowed to run, they check the difference between the first sample and the current time. If the delta is greater than or equal to the delay, then that Event has just happened. We first check if the state of the Serial Output is zero. We then jump to OStateS to start the outputting of the "Start Bit". Because any Serial Output timings must be rock solid, we use a trick in lines 101-116 that helps greatly. We check if we are within a certain amount of time BEFORE the deadline and then wait for the time to output another bit. This trick allows us to be within a certain ± amount of time within the expected time to output that bit. With this code, we are about <±8% accurate for the Serial Output. You can only use this trick on the most critical tasks, and only on one. In this section of code, we are constantly checking the delta of time from the "FIRST_RTCC_O" reading and the current reading of RTCC. When we are very close to the output time, we jump to line 117. If we are not even close to the proper time, we exit back to the main loop, so we can check the other timers and tasks. Now look at Figure 4 for a description of the Output Pulses, the "Bit units of Time", and the associated state numbers. Note that the activities are spread out over time.

The timer Events help to define the different states and their associated output activities. Each Event is handled in a very short, well-defined set of code as Task #1. Lines 117-131, are a quick state jump table. You need to break all Real-Time code into very short segments - in and then out. Each segment is just a few lines long. You do your activity, save status, and increment to the next state. Notice that OState0_7 code is used several times to output all 8 bits. The state variable is used also to count the number of bits already outputted. The time to the next outputting of a bit is calculatedand is adjusted to take out the accumulation of errors in lines 151-152. We make sure of a full "Stop Bit" length in the OStateE code. In the OStateL code, we reset the OState variable to zero, and tell the world that we are not outputting now in line 157. This is important because we use that bit (OState_B) to Signal that we need to protect the variable xmt_byte that changes over several states. We also use it to Signal that we are ready for another byte to output. Look at Task #4. See how it uses this Semaphore to full advantage. We have just explained a Critical Segment variable as outlined in the theory sections of this article.

© 1994 Microchip Technology Inc.

Task #2 reads the Serial Input line 1, running at 4800 Baud. The code structure is very similar to that of Task #1. See Figure 3. Notice that there are more states than the Serial Output Task #1. Once the "Start Bit" is detected, we half step into the "Start Bit" to see if it was a "False Start" or not. We then sample and store the incoming bits to form an 8-bit byte just like Task #1. We sample the "Stop Bit" to see if it is a "Frame Error". We delay another 1/2 bit to get to the end of the "Stop Bit" if there was an "Frame Error" before resetting Task #1's state to 0. Otherwise, we reset Task #1's state to 0, and Signal that we are ready for another "Start Bit". The just received byte is stored in variable "RCV_Storage". A check is made to see if we already sent out the last received byte before clobbering the old byte with the new byte.

Task #3 reads the Serial Input line 2, running at 4800 Baud. The code structure is the same as Task #2. See Figure #3. The received byte is also put into the same storage variable as Task #2 - "RCV_Storage". When either Task #2 or Task #3 receives a valid byte, Task #8's counter is reset. You can up the Baud rate of Task #2 and 3 if you lower the output Baud rate of Task #1. Note that for reading the Serial Input Lines, you can be off by ±15% for each sampling, but not accumulatively.

Task #4 finds the next buffered byte to send out through Task #1. Task #4 also controls the order of which byte goes first over another less important byte of data. It can be said that Task #1 Blocks Task #4 from running. You can think of the Serial Output Line as a Shared Resource. The use of Semaphores here allow the Synchronization of data and actions.

Task #5 monitors the Level Reset Input Line and will reset the LED state variable if the line ever goes low. This task is always in the Ready State. This task is said to simply "pole the input line" when ever it can.

Task #6 debounces the seven sensor input lines, running every 20 mSecs. The variable "T_20_mS_CO" is incremented every 512 uSecs (Clock Tick) and is compared to the count needed to equal 20 mSecs. If it is time, the subroutine QCheck_T123 is called to see if Tasks #1-3 are in the Ready State. If any of the Tasks #1-3 are ready, they are ran and we then continue with Task #6. We compare the current value of the input Port_B to see if it stayed the same from the last reading 20 mSecs back. If the two readings are the same, then Port_B is considered to be stable and the possibly new value is placed in the variable "Old_RB" to be outputted by Task #1. The subroutine D_H_E_L is called to determine the new LED state. We then check if Task #1 was too busy to output the last sensor status byte, if so then that error is recorded.

FIGURE 3: SERIAL INPUT STATES VS. TIME DIAGRAM

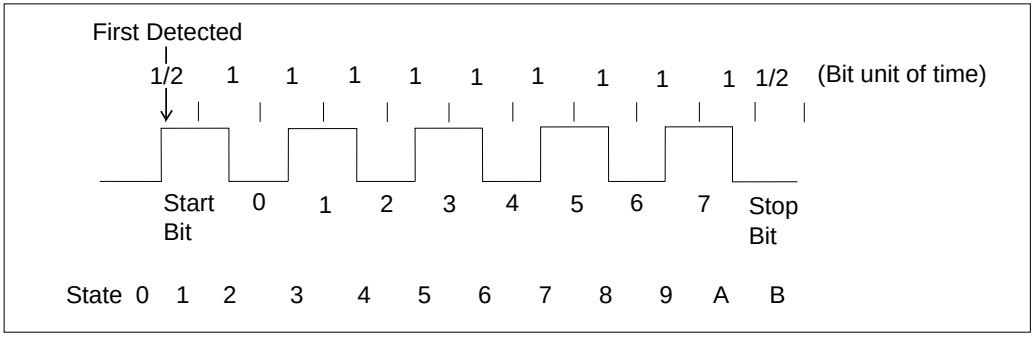
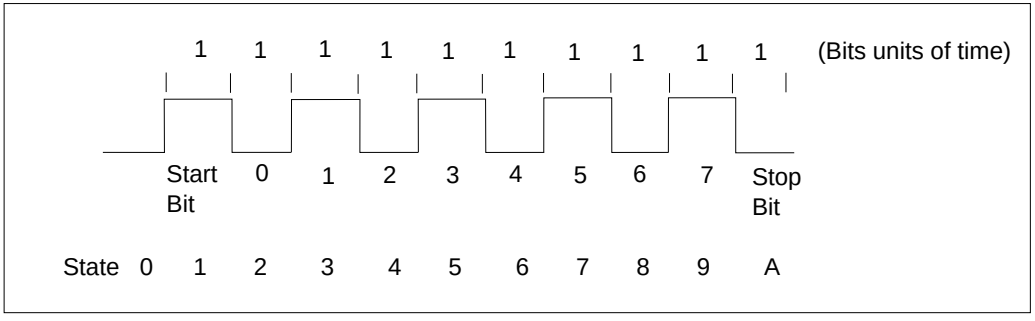


FIGURE 4: SERIAL OUTPUT STATES VS. TIME DIAGRAM



© 1994 Microchip Technology Inc.

Task #7 outputs the Highest Severity Level Indication on the LED. Do_LED starts at line 161, and continues to 276. This task is also broken into small time units of code. It is constantly checking to see if it is time to switch the on/off condition of the LED. The time units for Task #7 are regulated by the code in lines 613-619. 131072µS = time unit for Tasks #7-9. Task #7 has many state jump tables so it is included in the first 256 bytes of the first page. Lines 168-175 explain the on and off sequences and offs that represent levels of severity of the input status lines. The variable "LED_Mode" has both Task #7's current state number and the sub-state-number for that state's output sequence.

Task #8 is a 5 second lack of input from either of the two Serial input timers. Tasks #2 and #3 will reset the time counter for Task #8, when either receives a full byte. If the time counter "T_5_S_CO" equals 5 secs, then the LED's state is bumped to the highest, and a special byte is sent down the line to the next "Remote Alarm" unit. The counter variable is reset, and count starts all over. We then check if Task #1 was too busy to output the last special status byte, if so then that error is recorded.

Task #9 measures 5 minutes of calm on the 7 sensor lines and then resets the LED's state. Task #9 needs 16 bits of counter power to record 5 minutes of time. The counter variables are reset after being triggered.

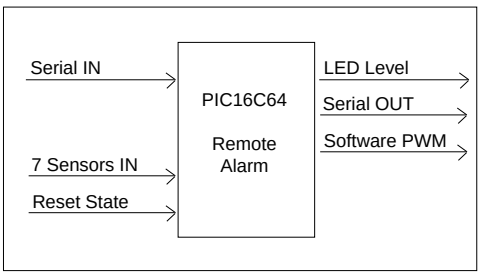
Do_D_H_E_L determines the LED's next state based on the 7 sensor input status. This subroutine checks each bit to see if it is active and then checks if a change in the LED's current state needs changing.

Do_Clear_Regs clears registers 7-1Fh. It leaves the FSR register zeroed out. This is very important for the PIC16C57 chip.

PIC16C64 - "Remote Alarm64" Example

This example is the same as the PIC16C54 example with a few changes to take advantage of the three timers on the PIC16C64 and interrupts. The second Serial input routine was replaced by an example of a software PWM (Pulse Width Modulation) example. The same code as the PIC16C54 example will run on the PIC16C64 with very few changes using only the TMR0 (RTCC). Be sure to read about the PIC16C54 example, as the comments will not be repeated, except to make a strong point.

FIGURE 5: REMOTE ALARM - PIC16C64 EXAMPLE



This example operates at 4Mhz. By simply increasing the crystal speeds, you can change the input and output Baud rates just as outlined in the section on the PIC16C54 example's crystal selection. By adjusting the constants in the code for the Serial routines, other Baud rates can be achieved at other crystal speeds. Note that you must use a very stable crystal setup and NOT an RC combination to run these examples.

We will now give a quick outline of the PIC16C64 code example. Lines 1-78 are the equates for this program. Notice that there is no need for jump tables for subroutines to be in the "first 256 bytes" of each page as there was in the PIC16C54 example. Note that the "Reset Vector" is now at code address 0, and the "Interrupt Vector" is at code address 4. Task #1 and 2 have been simplified greatly by using interrupts and timers. For Task #1, we no longer need to use the "trick" any more. It is time to execute once the routines for Task #1 and others are called. The section of code that handles the "Start Bit" (OStateS) lines 106-122 has been changed to setting up TMR2 with its interrupt to trigger the next call to this subroutine. The initial call to this subroutine was by Task #4, but later calls are due to Timer 2's interrupts. The amount of time until the next interrupt is set byeach state's code. This amount is based on the "Bit Unit of Time" which is based on Baud rate and crystal speed. An easy change to the code is to add a software selectable and "changeable on the fly" Baud rate. This is done by having a variable that selects the new Baud rate from the two data tables. One table gets you the Bit Delay value - see line 110. The other data table gets the value to be put into T2CON - see line 107, which selects the Post and Pre-scalers. You may need to adjust the Bit Delay value to take in account the Interrupt Latency. The OStateL state code shuts down Timer 2 and its interrupt. See lines 647-676 to understand how we get here by interrupt. Once Timer 2's count equals the count we put into register PR2, we get an interrupt if the following three conditions are true: 1.) Not already in an interrupt. When the current interrupt is done, our interrupt will be executed. 2.) GIE and PEIE bits are set. 3.) TMR2IE bit is set. Remember to clear the Flag bit as in line 114 before returning from an interrupt. Return from this subroutine will return you back to Task #4 or back to the ISR handle lines 647-676 depending on who called this routine. The Task #7's subroutines are the same as in the PIC16C54 example, lines 151-268. Task #2 is different from the previous example, lines 288-380. First Task #2 uses two interrupts. The INT interrupt on pin RB0/INT is used to detect the "Start Bit". It is very accurate. It is turned off after the detection in I1StateS code. The second interrupt TMR1 is then Enabled in the I1StateS code. Timer 1 is then used to cause an interrupt for all the other states for Task #2. Notice that Timer 1 has a 16-bit counter and we calculate the amount of Clock Ticks until overflow in lines 329-333. In the state code I1StateL, TMR1 is shut down, and the INT interrupt is now Enabled so as to detect the next input byte. The initializing of the PIC16C64 variable takes place in lines 383-426. The initializing of the tasks take place in lines 427-451. Notice that the last bit to be set

© 1994 Microchip Technology Inc.

is the GIE bit in line 451 after ALL is setup. There are several ways to execute the Task #3-9 code: by Timer 0 overflow interrupt, by having the code be in the background as in this example. The tradeoffs are many, and too deep for this article. Notice that the subroutine QCheck_T123 is not needed in this method. Timer0 overflow interrupt sets the flag: Time_Bit. The code in lines 454-457 can be considered the "IDLE Task" on some systems. It waits for a Clock Tick from TMR0's overflow. Task #3 is new, and is a simple 8-bit software PWM. Lines 459-478 show how to have 8 bits of ON, and 8 bits of OFF. This task has two states, on and off. You may add to the code by allowing the Real-Time changing of the 8-bit values under software control. When you change the values in the variables PWM_Out and PWM_In, disable all interrupts by using the following line: "BCF INTCON,GIE", and enable all interrupts by using the following line: "BSF INTCON,GIE". The new values will be used at the next transition, thus allowing a smooth change. This task could easily be used in the PIC16C54 example type Kernel. Task #4 is the same except that it calls Task #1's subroutine to initiate the outputting of a byte. See line 503. Tasks #5-9 are the same as in the PIC16C54 example. The subroutines: D_H_E_L and Clear_Regs are the same in both examples. The TMR0 (Timer 0) Overflow interrupt ISR (Interrupt Service Routine) is lines 641-645. This ISR will set the Time_Bit bit and clear the Flag that caused the interrupt. The Interrupt code lines 647-676 handles the saving of the Context Registers and the restoring of the Context Registers (W, Status, FSR) and by checking the order which interrupts are to be handled first - see lines 656-669. A very important line is 654. You must set the memory page pointers here for the ISR routines ! Line 676 is the only place that an interrupt is allowed to return and set the GIE bit (RETFIE).

Reentrant example

See Appendix E for the short code segment. This code corresponds to lines 302-332 in the PIC16C54 example. The purpose of reentrant code is to allow two or more tasks to use the same code at the "same time". See the section about reentrant in the theory section of this article. Notice how the registers 18h-1Bh match the registers 1Ch-1Fh, both starting with the state variable for the two tasks using this routine. Note how Task #2 and Task #3 load a pointer to the state variable for their task before calling DO_IState code. By using the FSR register as a pointer, and incrementing or decermenting the FSR register, you can keep the variables in the two tasks straight even if the two tasks are using different code in the subroutine at any one time. This method is not easy to implement, as can be seen, so use two copies for readability instead, like the PIC16C54 example.

SUMMARY

Now that the PIC16/17 family of microcontrollers have a way of executing Real-Time Programs, using the techniques outlined in this article, there is very little that PIC16/17s can not do! Much more than was ever dreamed before. Many of you will quickly understand and start modifying these examples. Great. That means that we have done our job at Myriad. A few of you may want more help. Great. At Myriad Development Co., we LOVE the PIC16/17 family.

© 1994 Microchip Technology Inc.

BIBLIOGRAPHY

Foster, Caxton C.
Real Time Programming - Neglected Topics
Reading, Massachusetts
Addison-Wesley Publishing Company, 1981

Holt, R.C., Graham, G.S., Lazowska, E.D., Scott, M.A.
Structured CONCURRENT PROGRAMMING with Operating Systems Applications
Reading, Massachusetts
Addison-Wesley Publishing Company, 1978

Kaisler, Stephen H.
The Design of Operating Systems for Small Computer Systems
New York, NY
John Wiley & Sons, 1983

Labrosse, Jean J.
uC/OS - The Real-Time Kernel
Lawrence, Kansas
R & D Publications, 1992

Loeliger, R.G.
Threaded Interpretive Languages
Peterborough, NH
BYTE BOOKS, 1981

© 1994 Microchip Technology Inc.

APPENDIX A

A Real-Time Vocabulary

ASYNCHRONOUS - an activity that can happen at any moment, at any time

BLOCKING - the act of wanting to waiting for an EVENT before continuing

CLOCK TICK - the heart beat that all time is based on

CONTEXT/TASK SWITCH - module that saves and restores the states of a task

CRITICAL SECTION - section of code or hardware - only one user at a time

DEADLOCK - that is where two TASKs are waiting for each others resources

DISPATCHING - the act of starting up a TASK to run from an RT Kernel

DYNAMIC PRIORITIES - the ability for TASKs to have there PRIORITIES changed

DYNAMIC TASKING - the creation and the killing of TASKs

EMBEDDED SYSTEM - an internal system that operates all by itself

ENABLING/DISABLING INTERRUPTS - controlling the interrupting processing

EVENT - Timer, communication, handshaking, interrupts, data, external events

EVENT FLAGS - the storage of current states or info on what has happened

INTERRUPT - a hardware event (external/internal) that triggers a jump to the ISR routines to handle that event

INTERRUPT LATENCY - how long it takes once signaled to start an ISR

INTERRUPT RECOVERY - how long it takes once interrupted to return back to code

KERNEL - module that controls TASKs, INTERRUPTs, and intertask communications

MAILBOXES - a way to pass data from one TASK to another

MASKABLE INTERRUPTS - the ability to control whether an ISR is called or not

MULTITASKING - the act of several TASKs thinking they own the CPU

MUTUAL EXCLUSION - the act of allowing only ONE owner to a RESOURCE

NMI - Non-Maskable Interrupt - can not be turned off by software

READY STATE - Referring to a list of TASKs ready (having work to do NOW)

REENTRANT - code that can be used by several TASKs at the same time

RESOURCE - data structures, display, I/O hardware, non-reentrant routines

RUNNING STATE - Referring to the ONE task owning/using the CPU currently

SCHEDULER - that part of a kernel that decides which TASK to run next

SEMAPHORES - a protocol to control RESOURCES, SIGNAL EVENTS, synchronize tasks

SIGNAL - the act of one task signaling another that something has happened

STATE MACHINE - an important concept in dividing a job into TASKs & ISRs

SYNCHRONIZATION - were TASKs synchronize over data or at a special time

TASK PRIORITY - each TASK is ranked as to its importance to getting done

TASK/THREAD - code that is defined by a small coherent job/work to be done

TIME SLICING - the act of giving the same amount of “time” to each TASK to run

TRAP - a software caused interrupt, useful for system access

WAITING STATE - Referring to a list of TASKs waiting for an EVENT(s)

© 1994 Microchip Technology Inc.

APPENDIX B

```
; 'Reentrant Code Example' Designed by Myriad Development Co. - Jerry Farmer
;       PIC16C54, 4MHz Crystal, WatchDog Timer OFF

; Register Files
IState1         equ     18h             ;Serial In #1 State
First_RTCC_I1   equ     19h             ;Starting time for next #1 Input event
nbti1           equ     1Ah             ;Next Bit #1 In Time - variable time
rcv_byte_1      equ     1Bh             ;Receive Serial #1 In byte
IState2         equ     1Ch             ;Serial In #2 State
First_RTCC_I2   equ     1Dh             ;Starting time for next #2 Input event
nbti2           equ     1Eh             ;Next Bit #2 In Time - variable time
rcv_byte_2      equ     1Fh             ;Receive Serial #2 In byte


;******         ;Task 2,3 - Asynchronous 2400 Baud Serial Input (LOW=0)
Do_IState
                movf    INDIR           ;if IState2 == 0
                btsc    STATUS,Z        ;               then Do Start Bit
                goto    IStateS
                movf    RTCC,W          ;Get current time
                movwf   temp
                incf    FSR             ;Point to First_RTCC_I(1,2)
                movf    INDIR,W         ;Get elapsed time; Time Unit = 2 uS
                subwf   temp
                incf    FSR             ;Point to nbti(1,2)
                movf    INDIR,W         ;Past time for next input bit ?
                subwf   temp,W
                btss    STATUS,0
                goto    L1
L0
                movlw   2               ;Point to IState(1,2)
                subwf   FSR
                movf    INDIR,W         ;Get (0-B) mode #
                andlw   H'0F'           ;Get only mode #
                addwf   PC              ;jump to subroutine

                goto    IStateS         ;Serial Start Bit
                goto    IStateS2        ;1/2 of Start Bit - see if False Start
                goto    IState0_7       ;Bit 0
                goto    IState0_7       ;Bit 1
                goto    IState0_7       ;Bit 2
                goto    IState0_7       ;Bit 3
                goto    IState0_7       ;Bit 4
                goto    IState0_7       ;Bit 5
                goto    IState0_7       ;Bit 6
                goto    IState0_7       ;Bit 7
                goto    IStateE         ;Serial Stop Bit
                goto    IStateL         ;Last State
L1
                clrf    FSR             ;Clear the FSR register
                retlw   0

;*****
Task_2          ;Task 2 - Asynchronous 2400 Baud Serial Input (LOW=0)
                movlw   IState1         ;Point to IState1
                movwf   FSR
                call    Do_IState
;*****
Task_3          ;Task 3 - Asynchronous 2400 Baud Serial Input (LOW=0)
                movlw   IState2         ;Point to IState2
                movwf   FSR
                call    Do_IState

                END
```


© 1994 Microchip Technology Inc.

```
MPASM 01.00.01 Inter.     APP_C.ASM   7-29-1994  11:18:16                PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT

                     0001 ; 'Remote Alarm' V1.02 Designed by Myriad Development Co. - Jerry Farmer
                     0002 ;       PIC16C54, 4MHz Crystal, WatchDog Timer OFF, MPASM instruction set
                     0003
                     0004         list    p=16C54,t=ON,c=132
                     0005
                     0006         include "PICREG.H"
                     0001 ;*************************  PIC16C5X Header     ************************
01FF                 0002 PIC54   equ     1FFH    ; Define Reset Vectors
01FF                 0003 PIC55   equ     1FFH
03FF                 0004 PIC56   equ     3FFH
07FF                 0005 PIC57   equ     7FFH
                     0006 ;
0001                 0007 RTCC    equ     1
0002                 0008 PC      equ     2
0003                 0009 STATUS  equ     3       ; F3 Reg is STATUS Reg.
0004                 0010 FSR     equ     4
                     0011 ;
0005                 0012 Port_A  equ     5
0006                 0013 Port_B  equ     6       ; I/O Port Assignments
0007                 0014 Port_C  equ     7
                     0015 ;
                     0016 ;*****************************************************************
                     0017 ;
                     0018 ;                        ; STATUS REG. Bits
0000                 0019 CARRY   equ     0       ; Carry Bit is Bit.0 of F3
0000                 0020 C       equ     0
0001                 0021 DCARRY  equ     1
0001                 0022 DC      equ     1
0002                 0023 Z_bit   equ     2       ; Bit 2 of F3 is Zero Bit
0002                 0024 Z       equ     2
0003                 0025 P_DOWN  equ     3
0003                 0026 PD      equ     3
0004                 0027 T_OUT   equ     4
0004                 0028 TO      equ     4
0005                 0029 PA0     equ     5
0006                 0030 PA1     equ     6
0007                 0031 PA2     equ     7
                     0032 ;
0001                 0033 Same    equ     1
0000                 0034 W       equ     0
                     0035 ;
0000                 0036 LSB     equ     0
0007                 0037 MSB     equ     7
```


© 1994 Microchip Technology Inc.

```
0038 ;
0039 TRUE    equ    1
0040 YES     equ    1
0041 FALSE   equ    0
0042 NO      equ    0
0043 ;
0044 ;*****************************************************************************
0006
0007
0008 ; Constants
0009 INDIR       equ    0              ;Indirect Register
0010 OUT_BIT_TIME equ    33h            ;9600 Baud, 104uS Bit Rate
0011 IN_BIT_TIME  equ    64h            ;4800 Baud, 208uS Bit Rate
0012 FUDGE_TIME   equ    23h            ;Current Time within a Fudge Factor
0013
0014 ; B Register Definitions
0015 #define Level_Reset    Port_B,0     ;Low will cause Past Level to reset
0016                                      ;RB.7 - RB.1 == Input from Sensors
0017 RB_TRIS      equ    B'11111111'    ;RB TRIS at INIT State == all input
0018 RB_MASK      equ    B'00000000'    ;What is High/Low for RB at INIT State
0019
0020 ; A Register Definitions - Programmable Inputs
0021 #define Serial_IN_1    Port_A,0     ;Serial Input #1 - 8 bits
0022 #define LED            Port_A,1     ;LED Output - Level/State Indicator
0023 #define Serial_Out     Port_A,2     ;Serial Output - 8 bits + passwords
0024 #define Serial_IN_2    Port_A,3     ;Serial Input #2 - 8 bits
0025
0026 RA_TRIS       equ    B'11111001'    ;RA TRIS at INIT State
0027 RA_MASK       equ    B'00000000'    ;What is High/Low for RA at INIT State
0028
0029 ; Register Files
0030 temp          equ    07h            ;Tempary holding register - PIC16C54/56
0031 Timer_Bits    equ    08h            ;Indicates which Timer(s) are Active = 1
0032 Flags         equ    09h            ;Error Flags
0033 LED_Mode      equ    0Ah            ;(0-2)=Mode, 3=LED_B, (4-6)=Seq #, 7=NEW
0034 OState        equ    0Bh            ;Serial Out State
0035 T_5_M_LO      equ    0Ch            ;5 Min Timer Counter - Low
0036 T_5_M_HI      equ    0Dh            ;5 Min Timer Counter - High
0037 T_5_S_CO      equ    0Eh            ;5 Second Timer - lack of Serial Input
0038 T_20_mS_CO    equ    0Fh            ;20 mS Timer - used for debouncing
0039 LED_C         equ    10h            ;LED Counter
0040 Last_RTCC     equ    11h            ;Last value of the RTCC
0041 First_RTCC_O  equ    12h            ;Starting time for next Output event
0042 xmt_byte      equ    13h            ;Serial xmit byte - destroyed in use
0043 cc            equ    14h            ;256 * RTCC time
0044 RCV_Storage   equ    15h            ;Long term storage of rcv_byte #1 & 2
0045 Old_RB        equ    16h            ;Oldest/Master copy of RB
0046 Last_RB       equ    17h            ;Last copy of RB
```

```
0018                0047 IState1         equ     18h             ;Serial In #1 State
0019                0048 First_RTCC_I1   equ     19h             ;Starting time for next #1 Input event
001A                0049 nbti1           equ     1Ah             ;Next Bit #1 In Time - variable time
001B                0050 rcv_byte_1      equ     1Bh             ;Receive Serial #1 In byte
001C                0051 IState2         equ     1Ch             ;Serial In #2 State
001D                0052 First_RTCC_I2   equ     1Dh             ;Starting time for next #2 Input event
001E                0053 nbti2           equ     1Eh             ;Next Bit #2 In Time - variable time
001F                0054 rcv_byte_2      equ     1Fh             ;Receive Serial #2 In byte
                    0055
                    0056 ; Indicates which Timer(s) are Active = 1 & Flags
0006                0057  #define OState_B        Timer_Bits,0    ;Serial Out Active Bit
0007                0058  #define IState1_B       Timer_Bits,1    ;Serial IN #1 Active Bit
0008                0059  #define IState2_B       Timer_Bits,2    ;Serial IN #2 Active Bit
0009                0060  #define T_5_S_B         Timer_Bits,3    ;5 Second Timer Active Bit
000A                0061  #define T_5_M_B         Timer_Bits,4    ;5 Min Timer Active Bit
000B                0062  #define RCV_Got_One_B   Timer_Bits,5    ;Got a NEW Received byte to send out
000C                0063  #define RB_NEW_B        Timer_Bits,6    ;Indicates a change in RB input
000D                0064  #define S_5_S_B         Timer_Bits,7    ;Serial In 5 secs of inactivity
                    0065
                    0066 ; Error Flags
000E                0067  #define FS_Flag_1       Flags,0         ;Serial #1 IN had a False Start Error
000F                0068  #define FE_Flag_1       Flags,1         ;Last Serial #1 IN had a Frame Error
0010                0069  #define FS_Flag_2       Flags,2         ;Serial #2 IN had a False Start Error
0011                0070  #define FE_Flag_2       Flags,3         ;Last Serial #2 IN had a Frame Error
0012                0071  #define RCV_Overflow    Flags,4         ;Lost Serial Input Byte - too Slow
0013                0072  #define RB_Overflow     Flags,5         ;Lost RB Input Byte - too Slow
0014                0073  #define S_5_S_Overflow  Flags,6         ;Lost '5S Inactivity' msg - too Slow
                    0074
                    0075 ;Equates for LED Task #7
0015                0076  #define LED_B           LED_Mode,3      ;LED is active
0016                0077  #define LED_NEW_B       LED_Mode,7      ;LED has just changed Modes = 1
0008                0078 LED_OFF_MODE    equ     B'00001000'     ;LED OFF
0089                0079 LED_SEQ1_MODE   equ     B'10001001'     ;LED Sequence 1: .2s On, 1s Off
008A                0080 LED_SEQ2_MODE   equ     B'10001010'     ;LED Sequence 2: 3x(.2s), 1s Off
008B                0081 LED_SEQ3_MODE   equ     B'10001011'     ;LED Sequence 3: 5x(.2s), 1s Off
009C                0082 LED_SLOW_MODE   equ     B'10011100'     ;LED Slow Pulsing - .3 Hz
009D                0083 LED_MEDIUM_MODE equ     B'10011101'     ;LED Medium Pulsing - 1 Hz
009E                0084 LED_FAST_MODE   equ     B'10011110'     ;LED Fast Pulsing - 3 Hz
008F                0085 LED_ON_MODE     equ     B'10001111'     ;LED ON Continuously
                    0086
                    0087
                    0088 ; Clear Registers 7-1Fh
                    0089 Clear_Regs
0000 0BE9           0090         GOTO    Do_Clear_Regs   ;Save space in first 256 bytes
                    0091
                    0092 ; Determine the Highest Error Level & Start Task #7 outputing the new Level
                    0093 D_H_E_L
0001 0BD2           0094         GOTO    Do_D_H_E_L      ;Save space in first 256 bytes
```


© 1994 Microchip Technology Inc.

```
                     0095
                     0096 ;******                          ;Task #1 - Asynchronous 9600 Baud Serial Output (LOW=0)
                     0097 Do_OState
0002 022B            0098          MOVF    OState          ;if OState == 0
0003 0643            0099          BTFSC   STATUS,Z        ;
0004 0A24            0100          GOTO    OStateS         ;           then goto Output-Start-Bit
0005 0201            0101          MOVF    RTCC,W          ;Get current time
0006 0027            0102          MOVWF   temp            ;                 & store in tempary variable
0007 0212            0103          MOVF    First_RTCC_O,W  ;Get elapsed time; Time Unit = 2 uS
0008 00A7            0104          SUBWF   temp            ;Delta of Current Time & Orginal Time
0009 0C23            0105          MOVLW   FUDGE_TIME      ;Take in account processing time to do it
000A 0087            0106          SUBWF   temp,W          ;Time within fudge factor ?
000B 0703            0107          BTFSS   STATUS,C
000C 0A23            0108          GOTO    _0005           ;Not time yet to change States so return
000D 0C33            0109 _0003    MOVLW   OUT_BIT_TIME    ;Past time for next out-bit ?
000E 0087            0110          SUBWF   temp,W
000F 0603            0111          BTFSC   STATUS,C        ;Do some delaying until it is time
0010 0A15            0112          GOTO    _0004           ;It is now time to out put a bit
0011 0C04            0113          MOVLW   H'04'           ;Account for loop delay
0012 01E7            0114          ADDWF   temp
0013 0000            0115          NOP                    ;   make loop delay even
0014 0A0D            0116          GOTO    _0003           ;Wait for exact time to output bit
0015 020B            0117 _0004    MOVF    OState,W        ;Get (0-A) mode #
0016 0E0F            0118          ANDLW   H'0F'           ;Get only mode #
0017 01E2            0119          ADDWF   PC              ;jump to subroutine
0018 0A24            0120          GOTO    OStateS         ;Serial Start Bit
0019 0A2B            0121          GOTO    OState0_7       ;Bit 0
001A 0A2B            0122          GOTO    OState0_7       ;Bit 1
001B 0A2B            0123          GOTO    OState0_7       ;Bit 2
001C 0A2B            0124          GOTO    OState0_7       ;Bit 3
001D 0A2B            0125          GOTO    OState0_7       ;Bit 4
001E 0A2B            0126          GOTO    OState0_7       ;Bit 5
001F 0A2B            0127          GOTO    OState0_7       ;Bit 6
0020 0A2B            0128          GOTO    OState0_7       ;Bit 7
0021 0A31            0129          GOTO    OStateE         ;Serial Stop Bit
0022 0A36            0130          GOTO    OStateL         ;Last State
0023 0800            0131 _0005    RETLW   H'00'
                     0132
                     0133 OStateS
0024 0545            0134          BSF     Serial_Out      ;Serial Start Bit
0025 0201            0135          MOVF    RTCC,W          ;Store starting time
0026 0032            0136          MOVWF   First_RTCC_O
0027 0C0D            0137          MOVLW   H'0D'           ;Fudge again
0028 00B2            0138          SUBWF   First_RTCC_O
0029 02AB            0139          INCF    OState          ;increment to next state
002A 0800            0140          RETLW   H'00'
                     0141
                     0142 OState0_7                       ;Bit 0 - 7
```

© 1994 Microchip Technology Inc.

```
002B 0333          0143          RRF     xmt_byte        ;Move bit into C from right most bit
002C 0703          0144          BTFSS   STATUS,C        ;
002D 0445          0145          BCF     Serial_Out      ;
002E 0603          0146          BTFSC   STATUS,C        ;
002F 0545          0147          BSF     Serial_Out      ;
0030 0A32          0148          GOTO    OS_End
                   0149 OStateE
0031 0445          0150          BCF     Serial_Out      ;Serial Stop Bit
0032 0C33          0151 OS_End   MOVLW   OUT_BIT_TIME    ;Adjust out the cumulation of error
0033 01F2          0152          ADDWF   First_RTCC_O
0034 02AB          0153          INCF    OState          ;increment to next state
0035 0800          0154          RETLW   H'00'
                   0155 OStateL
0036 006B          0156          CLRF    OState          ;Ready to send next byte out
0037 0408          0157          BCF     OState_B        ;Serial Out not active
0038 0800          0158          RETLW   H'00'
                   0159
                   0160 ;******          ;Task #7 - Output Highest Level Indication on LED
                   0161 Do_LED
0039 06EA          0162          BTFSC   LED_NEW_B       ;Initialize regs if change in modes
003A 0A4C          0163          GOTO    LED_NEW
003B 02B0          0164          INCF    LED_C           ;Inc Counter - Time Unit = 131072 uS
003C 020A          0165          MOVF    LED_Mode,W      ;Get (0-7) mode #
003D 0E07          0166          ANDLW   H'07'           ;Get only mode #
003E 01E2          0167          ADDWF   PC              ;jump to subroutine
003F 0A48          0168          GOTO    LED_OFF         ;LED OFF
0040 0A64          0169          GOTO    LED_SEQ1        ;LED Seq 1: 1 short pulse & pause
0041 0A67          0170          GOTO    LED_SEQ2        ;LED Seq 2: 2 short pulses & pause
0042 0A8A          0171          GOTO    LED_SEQ3        ;LED Seq 3: 3 short pulses & pause
0043 0A50          0172          GOTO    LED_SLOW        ;LED Slow Pulsing - .3 Hz
0044 0A5E          0173          GOTO    LED_MEDIUM      ;LED Medium Pulsing - 1 Hz
0045 0A61          0174          GOTO    LED_FAST        ;LED Fast Pulsing - 3 Hz
0046 0A4D          0175          GOTO    LED_ON          ;LED ON Continuously
0047 0800          0176 _0012    RETLW   H'00'
                   0177 ;___
                   0178 LED_OFF
0048 0425          0179          BCF     LED             ;Turn off LED
0049 046A          0180          BCF     LED_B           ;LED must be off
004A 0070          0181          CLRF    LED_C           ;Reset Counter - LED_C = 0
004B 0800          0182          RETLW   H'00'
                   0183 ;___
                   0184 LED_NEW
004C 04EA          0185          BCF     LED_NEW_B       ;Done initializing
                   0186 LED_ON
004D 0525          0187          BSF     LED             ;Turn on LED
004E 0070          0188          CLRF    LED_C           ;Reset Counter - LED_C = 0
004F 0800          0189          RETLW   H'00'
                   0190 ;___
```


© 1994 Microchip Technology Inc.

```
0191 LED_SLOW
0050 0C0C 0192          MOVLW   H'0C'           ;.3Hz @ 50% Duty
0051 0027 0193          MOVWF   temp
0052 0207 0194 LED_S    MOVF    temp,W          ;Check LED_C if time, .3Hz @ 50% Duty
0053 0090 0195          SUBWF   LED_C,W
0054 0743 0196          BTFSS   STATUS,Z
0055 0A47 0197          GOTO    _0012
0056 0C10 0198          MOVLW   H'10'
0057 01AA 0199          XORWF   LED_Mode        ;Switch states
0058 078A 0200          BTFSS   LED_Mode,4      ;Now make LED same state
0059 0425 0201          BCF     LED
005A 068A 0202          BTFSC   LED_Mode,4
005B 0525 0203          BSF     LED
005C 0070 0204          CLRF    LED_C           ;Reset LED_C
005D 0800 0205          RETLW   H'00'
          0206 ;___
          0207 LED_MEDIUM
005E 0C04 0208          MOVLW   H'04'           ;1Hz @ 50% Duty
005F 0027 0209          MOVWF   temp
0060 0A52 0210          GOTO    LED_S           ;Go do it
          0211 ;___
          0212 LED_FAST
0061 0C01 0213          MOVLW   H'01'           ;3Hz @ 50% Duty
0062 0027 0214          MOVWF   temp
0063 0A52 0215          GOTO    LED_S           ;Go do it
          0216 ;___
          0217 LED_SEQ1
0064 078A 0218          BTFSS   LED_Mode,4      ;Skip if bit is high
0065 0A76 0219          GOTO    ON1             ;Go do it
0066 0A82 0220          GOTO    OFF3            ;Go do it
          0221 ;___
          0222 LED_SEQ2
0067 020A 0223          MOVF    LED_Mode,W      ;.2 ON, .2 OFF, .2 ON, 1 OFF
0068 0027 0224          MOVWF   temp
0069 0C30 0225          MOVLW   H'30'           ;Get sequence # only
006A 0167 0226          ANDWF   temp
006B 03A7 0227          SWAPF   temp            ;swap nibbles
006C 0207 0228          MOVF    temp,W          ;get nibble for offset
006D 01E2 0229          ADDWF   PC              ;Table jump calculation
006E 0A76 0230          GOTO    ON1             ;LED is on, check if time to change
006F 0A7C 0231          GOTO    OFF2            ;LED is off, check if time to change
0070 0A76 0232          GOTO    ON1             ;LED is on, check if time to change
0071 0A82 0233          GOTO    OFF3            ;LED is off, check if time to change
          0234 ;___
          0235 LED_Exit
0072 0C10 0236          MOVLW   H'10'           ;Inc Seq #
0073 01EA 0237          ADDWF   LED_Mode
0074 0070 0238          CLRF    LED_C           ;Reset LED_C
```

5

```
0075 0800 0239                 RETLW   H'00'
          0240 ON1
0076 0C02 0241                 MOVLW   H'02'           ;Check LED_C if time, .2 sec-on
0077 0090 0242                 SUBWF   LED_C,W
0078 0743 0243                 BTFSS   STATUS,Z
0079 0A47 0244                 GOTO    _0012
007A 0425 0245                 BCF     LED             ;Turn off LED
007B 0A72 0246                 GOTO    LED_Exit
          0247 OFF2
007C 0C02 0248                 MOVLW   H'02'           ;Check LED_C if time, .2 sec-on
007D 0090 0249                 SUBWF   LED_C,W
007E 0743 0250                 BTFSS   STATUS,Z
007F 0A47 0251                 GOTO    _0012
0080 0525 0252                 BSF     LED             ;Turn on LED
0081 0A72 0253                 GOTO    LED_Exit
          0254 OFF3
0082 0C08 0255                 MOVLW   H'08'           ;Check LED_C if time, 1 sec-off
0083 0090 0256                 SUBWF   LED_C,W
0084 0743 0257                 BTFSS   STATUS,Z
0085 0A47 0258                 GOTO    _0012
0086 0525 0259                 BSF     LED             ;Turn on LED
0087 0CF0 0260                 MOVLW   H'F0'
0088 012A 0261                 IORWF   LED_Mode        ;Cause (Seq# & NEW) to overflow to 0
0089 0A72 0262                 GOTO    LED_Exit
          0263 LED_SEQ3                                ;.2 ON, .2 OFF, .2 ON, .2 OFF, .2 ON, 1 OFF
008A 020A 0264                 MOVF    LED_Mode,W      ;Get LED info
008B 0027 0265                 MOVWF   temp
008C 0C70 0266                 MOVLW   H'70'           ;Get sequence # only
008D 0167 0267                 ANDWF   temp
008E 03A7 0268                 SWAPF   temp            ;swap nibbles
008F 0207 0269                 MOVF    temp,W          ;get nibble for offset
0090 01E2 0270                 ADDWF   PC              ;Table jump calculation
0091 0A76 0271                 GOTO    ON1             ;LED is on check if time to change
0092 0A7C 0272                 GOTO    OFF2            ;LED is off check if time to change
0093 0A76 0273                 GOTO    ON1             ;LED is on check if time to change
0094 0A7C 0274                 GOTO    OFF2            ;LED is off check if time to change
0095 0A76 0275                 GOTO    ON1             ;LED is on check if time to change
0096 0A82 0276                 GOTO    OFF3            ;LED is off check if time to change
          0277
          0278 ;****                                   Quick Check of Tasks #1, #2 and #3
          0279 QCheck_T123
          0280                 ;Task #1 - Asynchronous 9600 Baud Serial Output (LOW=0)
0097 0708 0281                 BTFSS   OState_B        ;if not outputing now then skip call
0098 0A9A 0282                 GOTO    T2
0099 0902 0283                 CALL    Do_OState       ;Go Do Task #1
          0284
          0285                 ;Task #2 - Asynchronous 4800 Baud Serial Input (LOW=0)
009A 0628 0286 T2              BTFSC   IState1_B       ;if already started then call
```


© 1994 Microchip Technology Inc.

```
009B 0A9F           0287         GOTO    _0029
009C 0605           0288         BTFSC   Serial_IN_1     ;if Start bit ? then call
009D 0A9F           0289         GOTO    _0029
009E 0AA0           0290         GOTO    T3
009F 09A7           0291 _0029   CALL    Do_I1State      ;Go Do Task #2
                    0292
                    0293         ;Task #3 - Asynchronous 4800 Baud Serial Input (LOW=0)
00A0 0648           0294 T3      BTFSC   IState2_B       ;if already started then call
00A1 0AA5           0295         GOTO    _0031
00A2 0665           0296         BTFSC   Serial_IN_2     ;if Start bit ? then call
00A3 0AA5           0297         GOTO    _0031
00A4 0800           0298         RETLW   H'00'
00A5 09C2           0299 _0031   CALL    Do_I2State      ;Go Do Task #3
00A6 0800           0300         RETLW   H'00'
                    0301
                    0302 ;******         ;Task #2 - Asynchronous 4800 Baud Serial Input (LOW=0)
                    0303 Do_I1State
00A7 0238           0304         MOVF    IState1         ;if IState1 == 0
00A8 0643           0305         BTFSC   STATUS,Z        ;               then Do Start Bit
00A9 0ADD           0306         GOTO    I1StateS
00AA 0201           0307         MOVF    RTCC,W          ;Get current time
00AB 0027           0308         MOVWF   temp
00AC 0219           0309         MOVF    First_RTCC_I1,W ;Get elapsed time; Time Unit = 2 uS
00AD 00A7           0310         SUBWF   temp
00AE 021A           0311         MOVF    nbti1,W         ;Past time for next input bit ?
00AF 0087           0312         SUBWF   temp,W
00B0 0703           0313         BTFSS   STATUS,C
00B1 0AC1           0314         GOTO    _0033
00B2 0218           0315         MOVF    IState1,W       ;Get (0-B) mode #
00B3 0E0F           0316         ANDLW   H'0F'           ;Get only mode #
00B4 01E2           0317         ADDWF   PC              ;jump to subroutine
00B5 0ADD           0318         GOTO    I1StateS        ;Serial Start Bit
00B6 0AE6           0319         GOTO    I1State2        ;1/2 of Start Bit - see if False Start
00B7 0AEF           0320         GOTO    I1State0_7      ;Bit 0
00B8 0AEF           0321         GOTO    I1State0_7      ;Bit 1
00B9 0AEF           0322         GOTO    I1State0_7      ;Bit 2
00BA 0AEF           0323         GOTO    I1State0_7      ;Bit 3
00BB 0AEF           0324         GOTO    I1State0_7      ;Bit 4
00BC 0AEF           0325         GOTO    I1State0_7      ;Bit 5
00BD 0AEF           0326         GOTO    I1State0_7      ;Bit 6
00BE 0AEF           0327         GOTO    I1State0_7      ;Bit 7
00BF 0AF8           0328         GOTO    I1StateE        ;Serial Stop Bit
00C0 0B03           0329         GOTO    I1StateL        ;Last State - End of Stop Bit
                    0330 _0033
00C1 0800           0331         RETLW   H'00'
                    0332
                    0333 ;******         ;Task #3 - Asynchronous 4800 Baud Serial Input (LOW=0)
                    0334 Do_I2State
```

© 1994 Microchip Technology Inc.

```
00C2 023C 0335         MOVF    IState2         ;if IState1 == 0
00C3 0643 0336         BTFSC   STATUS,Z        ;           then Do Start Bit
00C4 0B10 0337         GOTO    I2StateS
00C5 0201 0338         MOVF    RTCC,W          ;Get current time
00C6 0027 0339         MOVWF   temp
00C7 021D 0340         MOVF    First_RTCC_I2,W ;Get elapsed time; Time Unit = 2 uS
00C8 00A7 0341         SUBWF   temp
00C9 021E 0342         MOVF    nbti2,W         ;Past time for next input bit ?
00CA 0087 0343         SUBWF   temp,W
00CB 0703 0344         BTFSS   STATUS,C
00CC 0ADC 0345         GOTO    _0035
00CD 021C 0346         MOVF    IState2,W       ;Get (0-B) mode #
00CE 0E0F 0347         ANDLW   H'0F'           ;Get only mode #
00CF 01E2 0348         ADDWF   PC              ;jump to subroutine
00D0 0B10 0349         GOTO    I2StateS        ;Serial Start Bit
00D1 0B19 0350         GOTO    I2StateS2       ;1/2 of Start Bit - see if False Start
00D2 0B22 0351         GOTO    I2State0_7      ;Bit 0
00D3 0B22 0352         GOTO    I2State0_7      ;Bit 1
00D4 0B22 0353         GOTO    I2State0_7      ;Bit 2
00D5 0B22 0354         GOTO    I2State0_7      ;Bit 3
00D6 0B22 0355         GOTO    I2State0_7      ;Bit 4
00D7 0B22 0356         GOTO    I2State0_7      ;Bit 5
00D8 0B22 0357         GOTO    I2State0_7      ;Bit 6
00D9 0B22 0358         GOTO    I2State0_7      ;Bit 7
00DA 0B2B 0359         GOTO    I2StateE        ;Serial Stop Bit
00DB 0B36 0360         GOTO    I2StateL        ;Last State - End of Stop Bit
00DC 0800 0361 _0035   RETLW   H'00'
          0362
          0363 ;***                            ;Subroutines for Task #2
          0364 I1StateS                        ;Start Bit - Setup timing variables
00DD 0528 0365         BSF     IState1_B       ;Serial Input Active
00DE 0201 0366         MOVF    RTCC,W          ;Store starting time
00DF 0039 0367         MOVWF   First_RTCC_I1
00E0 0C0D 0368         MOVLW   H'0D'           ;Fudge again
00E1 00B9 0369         SUBWF   First_RTCC_I1
00E2 0C32 0370         MOVLW   H'32'           ;Time delay = 1/2 bit time
00E3 003A 0371         MOVWF   nbti1
00E4 02B8 0372         INCF    IState1         ;Increment to next state
00E5 0800 0373         RETLW   H'00'
          0374 I1State2                        ;Check if still a Start Bit
00E6 0705 0375         BTFSS   Serial_IN_1     ;False Start Error ?
00E7 0B06 0376         GOTO    FS_Error_1
00E8 0409 0377         BCF     FS_Flag_1       ;Start Bit OK
00E9 021A 0378         MOVF    nbti1,W         ;Adjust out the error
00EA 01F9 0379         ADDWF   First_RTCC_I1
00EB 0C64 0380         MOVLW   IN_BIT_TIME     ;Time Delay = full bit time
00EC 003A 0381         MOVWF   nbti1
00ED 02B8 0382         INCF    IState1         ;increment to next state
```


© 1994 Microchip Technology Inc.

```
00EE 0800           0383         RETLW   H'00'
                    0384 I1State0_7                     ;Bit 0 - 7
00EF 0705           0385         BTFSS   Serial_IN_1    ;Move Input bit into C
00F0 0403           0386         BCF     STATUS,C
00F1 0605           0387         BTFSC   Serial_IN_1
00F2 0503           0388         BSF     STATUS,C
00F3 033B           0389         RRF     rcv_byte_1     ;Move C into left most bit
00F4 021A           0390         MOVF    nbti1,W
00F5 01F9           0391         ADDWF   First_RTCC_I1  ;Adjust out the error
00F6 02B8           0392         INCF    IState1        ;increment to next state
00F7 0800           0393         RETLW   H'00'
                    0394 I1StateE                       ;Check if we have a proper Stop Bit
00F8 0605           0395         BTFSC   Serial_IN_1    ;Frame Error
00F9 0B09           0396         GOTO    F_Error_1
00FA 0429           0397         BCF     FE_Flag_1      ;Stop Bit OK
00FB 006E           0398         CLRF    T_5_S_CO       ;Reset 5 Sec Timer - got a good byte
                    0399         ;Process the msg Here !
00FC 021B           0400         MOVF    rcv_byte_1,W   ;Make a copy of just received byte
00FD 0035           0401         MOVWF   RCV_Storage
00FE 07A8           0402         BTFSS   RCV_Got_One_B  ;Report Lost data
00FF 0489           0403         BCF     RCV_Overflow
0100 06A8           0404         BTFSC   RCV_Got_One_B
0101 0589           0405         BSF     RCV_Overflow
0102 05A8           0406         BSF     RCV_Got_One_B  ;We Now have a RB Value to go out
                    0407 I1StateL
0103 0078           0408         CLRF    IState1        ;Ready to receive next byte
0104 0428           0409         BCF     IState1_B      ;Serial In not currently active
0105 0800           0410         RETLW   H'00'
                    0411 FS_Error_1                     ;False Start - Shut Down Checking
0106 0428           0412         BCF     IState1_B      ;Serial Input NOT Active
0107 0509           0413         BSF     FS_Flag_1      ;False Start Error
0108 0B03           0414         GOTO    I1StateL       ;Start All Over
                    0415 F_Error_1                      ;Frame Error - Wait for End of Stop Bit
0109 021A           0416         MOVF    nbti1,W        ;Adjust out the error
010A 01F9           0417         ADDWF   First_RTCC_I1
010B 0C32           0418         MOVLW   H'32'          ;Time Delay = 1/2 bit time
010C 003A           0419         MOVWF   nbti1
010D 0529           0420         BSF     FE_Flag_1      ;Frame Error for this Byte ?
010E 02B8           0421         INCF    IState1        ;Increment to next state
010F 0800           0422         RETLW   H'00'
                    0423
                    0424 ;***            ;Subroutines for Task #3
                    0425 I2StateS                       ;Start Bit - Setup timing variables
0110 0548           0426         BSF     IState2_B      ;Serial Input Active
0111 0201           0427         MOVF    RTCC,W         ;Store starting time
0112 003D           0428         MOVWF   First_RTCC_I2
0113 0C0D           0429         MOVLW   H'0D'          ;Fudge again
0114 00BD           0430         SUBWF   First_RTCC_I2
```

© 1994 Microchip Technology Inc.

```
0115 0C32                0431          MOVLW   H'32'           ;Time delay = 1/2 bit time
0116 003E                0432          MOVWF   nbti2
0117 02BC                0433          INCF    IState2         ;Increment to next state
0118 0800                0434          RETLW   H'00'
                         0435 I2StateS2                        ;Check if still a Start Bit
0119 0765                0436          BTFSS   Serial_IN_2     ;False Start Error ?
011A 0B39                0437          GOTO    FS_Error_2
011B 0449                0438          BCF     FS_Flag_2       ;Start Bit OK
011C 021E                0439          MOVF    nbti2,W         ;Adjust out the error
011D 01FD                0440          ADDWF   First_RTCC_I2
011E 0C64                0441          MOVLW   IN_BIT_TIME     ;Time Delay = full bit time
011F 003E                0442          MOVWF   nbti2
0120 02BC                0443          INCF    IState2         ;increment to next state
0121 0800                0444          RETLW   H'00'
                         0445 I2State0_7                       ;Bit 0 - 7
0122 0765                0446          BTFSS   Serial_IN_2     ;Move Input bit into C
0123 0403                0447          BCF     STATUS,C
0124 0665                0448          BTFSC   Serial_IN_2
0125 0503                0449          BSF     STATUS,C
0126 033F                0450          RRF     rcv_byte_2      ;Move C into left most bit
0127 021E                0451          MOVF    nbti2,W
0128 01FD                0452          ADDWF   First_RTCC_I2   ;Adjust out the error
0129 02BC                0453          INCF    IState2         ;increment to next state
012A 0800                0454          RETLW   H'00'
                         0455 I2StateE                         ;Check if we have a proper Stop Bit
012B 0665                0456          BTFSC   Serial_IN_2     ;Frame Error
012C 0B3C                0457          GOTO    F_Error_2
012D 0469                0458          BCF     FE_Flag_2       ;Stop Bit OK
012E 006E                0459          CLRF    T_5_S_CO        ;Reset 5 Sec Timer - got a good byte
                         0460          ;Process the msg Here !
012F 021F                0461          MOVF    rcv_byte_2,W    ;Make a copy of just received byte
0130 0035                0462          MOVWF   RCV_Storage
0131 07A8                0463          BTFSS   RCV_Got_One_B   ;Report Lost data
0132 0489                0464          BCF     RCV_Overflow
0133 06A8                0465          BTFSC   RCV_Got_One_B
0134 0589                0466          BSF     RCV_Overflow
0135 05A8                0467          BSF     RCV_Got_One_B   ;We Now have a RB Value to go out
                         0468 I2StateL
0136 007C                0469          CLRF    IState2         ;Ready to receive next byte
0137 0448                0470          BCF     IState2_B       ;Serial In not currently active
0138 0800                0471          RETLW   H'00'
                         0472 FS_Error_2
0139 0448                0473          BCF     IState2_B       ;False Start - Shut Down Checking
013A 0549                0474          BSF     FS_Flag_2       ;False Start Error
013B 0B36                0475          GOTO    I2StateL        ;Start All Over
                         0476 F_Error_2                        ;Frame Error - Wait for End of Stop Bit
013C 021E                0477          MOVF    nbti2,W         ;Adjust out the error
013D 01FD                0478          ADDWF   First_RTCC_I2
```


© 1994 Microchip Technology Inc.

```
013E 0C32          0479         MOVLW   H'32'           ;Time Delay = 1/2 bit time
013F 003E          0480         MOVWF   nbti2
0140 0569          0481         BSF     FE_Flag_2       ;Frame Error for this Byte ?
0141 02BC          0482         INCF    IState2         ;Increment to next state
0142 0800          0483         RETLW   H'00'
                   0484
                   0485 ;******         ;Code Starting point
                   0486 Main
0143 0C00          0487         MOVLW   H'00'           ;What is High/Low for RA at INIT State
0144 0025          0488         MOVWF   Port_A
0145 0C00          0489         MOVLW   H'00'           ;What is High/Low for RB at INIT State
0146 0026          0490         MOVWF   Port_B
0147 0CF9          0491         MOVLW   H'F9'           ;RA TRIS at INIT State
0148 0005          0492         TRIS    5
0149 0CFF          0493         MOVLW   H'FF'           ;RB TRIS at INIT State
014A 0006          0494         TRIS    6
014B 0C00          0495         MOVLW   H'00'           ;RTCC/2
014C 0002          0496         OPTION
014D 0900          0497         CALL    Clear_Regs      ;Clear Registers 7-1F - Same Memory Page
014E 0061          0498         CLRF    RTCC            ;Start timers
                   0499
                   0500 ;Initialize Tasks
                   0501                                 ;Task #1 waits for byte to output
                   0502                                 ;Task #2 waits for Serial IN Start Bit
                   0503                                 ;Task #3 waits for Serial IN Start Bit
                   0504                                 ;Task #4 runs when Task 1 is Not
                   0505                                 ;Task #5 is always running
014F 0206          0506         MOVF    Port_B,W        ;Task #6 is Initialized here
0150 0036          0507         MOVWF   Old_RB
0151 0216          0508         MOVF    Old_RB,W        ;Make all the same initial value
0152 0037          0509         MOVWF   Last_RB
0153 05C8          0510         BSF     RB_NEW_B        ;Tell Task #4: RB byte ready to output
0154 0C08          0511         MOVLW   LED_OFF_MODE
0155 002A          0512         MOVWF   LED_Mode        ;Task #7 is Started
0156 0568          0513         BSF     T_5_S_B         ;Task #8 is Started here
0157 0588          0514         BSF     T_5_M_B         ;Task #9 is Started here
                   0515
                   0516 ; Handle Task & Timer activities - Main Loop
                   0517 Task_1          ;Task #1 - Asynchronous 9600 Baud Serial Output (LOW=0)
0158 0708          0518         BTFSS   OState_B        ;if not outputing now then skip call
0159 0B5B          0519         GOTO    Task_2
015A 0902          0520         CALL    Do_OState       ;Go Do Task #1
                   0521
                   0522 Task_2          ;Task #2 - Asynchronous 4800 Baud Serial Input (LOW=0)
015B 0628          0523         BTFSC   IState1_B       ;if already started then call
015C 0B60          0524         GOTO    _0053
015D 0605          0525         BTFSC   Serial_IN_1     ;if Start bit ? then call
015E 0B60          0526         GOTO    _0053
```

© 1994 Microchip Technology Inc.

```
                      0527          GOTO    Task_3
015F 0B61             0528 _0053    CALL    Do_I1State      ;Go Do Task #2
0160 09A7             0529
                      0530 Task_3             ;Task #3 - Asynchronous 4800 Baud Serial Input (LOW=0)
0161 0648             0531          BTFSC   IState2_B       ;if already started then call
0162 0B66             0532          GOTO    _0055
0163 0665             0533          BTFSC   Serial_IN_2     ;if Start bit ? then call
0164 0B66             0534          GOTO    _0055
0165 0B67             0535          GOTO    Task_4
0166 09C2             0536 _0055    CALL    Do_I2State      ;Go Do Task #3
                      0537
                      0538 Task_4             ;Task #4 - Finds next Buffered Byte to Send Out through Task 1
0167 0608             0539          BTFSC   OState_B        ;if outputing now then skip call
0168 0B7D             0540          GOTO    _0059
0169 07A8             0541          BTFSS   RCV_Got_One_B   ;Got a NEW Received byte to send
016A 0B70             0542          GOTO    _0057
016B 0215             0543          MOVF    RCV_Storage,W   ;Send just received byte
016C 0033             0544          MOVWF   xmt_byte
016D 04A8             0545          BCF     RCV_Got_One_B   ;Clear need to send old byte
016E 0508             0546          BSF     OState_B        ;Start Task #1 & Lock Out Others
016F 0B7D             0547          GOTO    _0059
0170 07C8             0548 _0057    BTFSS   RB_NEW_B        ;Indicates a change in RB input
0171 0B77             0549          GOTO    _0058
0172 0216             0550          MOVF    Old_RB,W        ;Send New RB value
0173 0033             0551          MOVWF   xmt_byte
0174 04C8             0552          BCF     RB_NEW_B        ;Clear need to send out newest value
0175 0508             0553          BSF     OState_B        ;Start Task #1 & Lock Out Others
0176 0B7D             0554          GOTO    _0059
0177 07E8             0555 _0058    BTFSS   S_5_S_B         ;Serial In 5 secs of inactivity
0178 0B7D             0556          GOTO    _0059
0179 0CFF             0557          MOVLW   H'FF'           ;Tell of inactivity of Serial In
017A 0033             0558          MOVWF   xmt_byte
017B 04E8             0559          BCF     S_5_S_B         ;Clear need to send msg
017C 0508             0560          BSF     OState_B        ;Start Task #1 & Lock Out Others
                      0561
                      0562                  ;Heart Beat - Time unit = 512 uS for Tasks #5 & #6
017D 0201             0563 _0059    MOVF    RTCC,W          ;Step-up time units * 512
017E 0027             0564          MOVWF   temp
017F 0211             0565          MOVF    Last_RTCC,W     ;Test to see if it overflowed
0180 0087             0566          SUBWF   temp,W
0181 0703             0567          BTFSS   STATUS,C
0182 0B86             0568          GOTO    Inc_Time
0183 0207             0569          MOVF    temp,W          ;unit error = < |+-512 uS|
0184 0031             0570          MOVWF   Last_RTCC
0185 0B58             0571          GOTO    Task_1
                      0572 Inc_Time
0186 0207             0573          MOVF    temp,W          ;Save current RTCC into Last_RTCC
0187 0031             0574          MOVWF   Last_RTCC
```


© 1994 Microchip Technology Inc.

```
                        0575
                        0576 Task_5         ;Task #5 - Monitor Level Reset Input Line - Always Running !
0188 0606               0577         BTFSC  Level_Reset
0189 0B8C               0578         GOTO   Task_6
018A 0C08               0579         MOVLW  LED_OFF_MODE    ;Lowest Level Indicator output
018B 002A               0580         MOVWF  LED_Mode
                        0581
                        0582 Task_6         ;Task #6 - Debounce 8 bit Input Sensors - Runs every 20 mS
018C 02AF               0583         INCF   T_20_mS_CO      ;Inc Counter - Time Unit = 512 uS
018D 0C27               0584         MOVLW  H'27'           ;Used to debounce the input
018E 008F               0585         SUBWF  T_20_mS_CO,W
018F 0743               0586         BTFSS  STATUS,Z
0190 0BA7               0587         GOTO   _0065
0191 006F               0588         CLRF   T_20_mS_CO      ;Reset T_20_mS_CO to start over again
                        0589
0192 0997               0590         CALL   QCheck_T123     ;Quick Check of Tasks #1, #2 and #3
                        0591
0193 0206               0592         MOVF   Port_B,W        ;Last copy of RB same as Current ?
0194 0097               0593         SUBWF  Last_RB,W
0195 0643               0594         BTFSC  STATUS,Z
0196 0B9A               0595         GOTO   _0062
0197 0206               0596         MOVF   Port_B,W        ;  Store Current RB - diff from Last
0198 0037               0597         MOVWF  Last_RB
0199 0B9C               0598         GOTO   _0063
019A 0217               0599 _0062   MOVF   Last_RB,W       ;New Old RB <- same value over 20 mS
019B 0036               0600         MOVWF  Old_RB
019C 0236               0601 _0063   MOVF   Old_RB          ;See if RB is now 0
019D 0643               0602         BTFSC  STATUS,Z        ;RB == 0 ? then keep timer running
019E 0BA1               0603         GOTO   _0064
019F 006C               0604         CLRF   T_5_M_LO        ;Reset 5 Min Timer
01A0 006D               0605         CLRF   T_5_M_HI        ;  still not zero yet
01A1 0901               0606 _0064   CALL   D_H_E_L         ;Determine the Highest Error Level
01A2 07C8               0607         BTFSS  RB_NEW_B        ;Check for Lost Data Error
01A3 04A9               0608         BCF    RB_Overflow
01A4 06C8               0609         BTFSC  RB_NEW_B
01A5 05A9               0610         BSF    RB_Overflow
01A6 05C8               0611         BSF    RB_NEW_B        ;Every 20 mS send Old_RB out
                        0612
                        0613         ;Heart Beat - Time unit = 131072 uS for Tasks #7, #8 & #9
01A7 0CF9               0614 _0065   MOVLW  H'F9'           ;RA TRIS - refresh
01A8 0005               0615         TRIS   5
01A9 0CFF               0616         MOVLW  H'FF'           ;RB TRIS - refresh
01AA 0006               0617         TRIS   6
01AB 02F4               0618         DECFSZ cc              ;Step-up time units * 256
01AC 0B58               0619         GOTO   Task_1
                        0620
                        0621 Task_7         ;Task 7 - Output Highest Level Indication on LED
01AD 076A               0622         BTFSS  LED_B           ;Is LED active ?
```

© 1994 Microchip Technology Inc.

```
01AE 0BB1           0623         GOTO    Task_8
                    0624
01AF 0997           0625         CALL    QCheck_T123     ;Quick Check of Tasks #1, #2 and #3
                    0626
01B0 0939           0627         CALL    Do_LED          ;Handle LED timing
                    0628
                    0629 Task_8          ;Task #8 - 5 Second Serial Input Lack of Activity Timer
01B1 0768           0630         BTFSS   T_5_S_B         ;5 Sec Timer Active ?
01B2 0BC0           0631         GOTO    Task_9
01B3 02AE           0632         INCF    T_5_S_CO        ;Inc Counter - Time Unit = 131072 uS
01B4 0C26           0633         MOVLW   H'26'           ;Check T_5_S_CO if time
01B5 008E           0634         SUBWF   T_5_S_CO,W
01B6 0743           0635         BTFSS   STATUS,Z
01B7 0BC0           0636         GOTO    Task_9
01B8 006E           0637         CLRF    T_5_S_CO        ;Reset T_5_S_CO
01B9 0C8F           0638         MOVLW   LED_ON_MODE     ;Highest Level Indicator output
01BA 002A           0639         MOVWF   LED_Mode
01BB 07E8           0640         BTFSS   S_5_S_B         ;Check if Lost Data Error
01BC 04C9           0641         BCF     S_5_S_Overflow
01BD 06E8           0642         BTFSC   S_5_S_B
01BE 05C9           0643         BSF     S_5_S_Overflow
01BF 05E8           0644         BSF     S_5_S_B          ;Send notice of 5 seconds of inaction
                    0645
                    0646 Task_9          ;Task #9 - 5 Min. Lack of Severe Error from Sensors Reset Timer
01C0 0788           0647         BTFSS   T_5_M_B         ;5 Min Timer Active ?
01C1 0BD1           0648         GOTO    Task_A
01C2 02AC           0649         INCF    T_5_M_LO        ;Inc LO Counter; Time Unit = 131072 uS
01C3 0643           0650         BTFSC   STATUS,Z        ;See if carry needs to be passed on ?
01C4 02AD           0651         INCF    T_5_M_HI        ;Inc HI Counter; Time Unit = 131072 uS
01C5 0C08           0652         MOVLW   H'08'           ;#2288<  Check T_5_M_HI if time
01C6 008D           0653         SUBWF   T_5_M_HI,W
01C7 0743           0654         BTFSS   STATUS,Z
01C8 0BD1           0655         GOTO    Task_A
01C9 0CF0           0656         MOVLW   H'F0'           ;#2288>  Check T_5_M_LO if time
01CA 008C           0657         SUBWF   T_5_M_LO,W
01CB 0743           0658         BTFSS   STATUS,Z
01CC 0BD1           0659         GOTO    Task_A
01CD 006C           0660         CLRF    T_5_M_LO        ;Reset T_5_M_LO
01CE 006D           0661         CLRF    T_5_M_HI        ;Reset T_5_M_HI
01CF 0C08           0662         MOVLW   LED_OFF_MODE    ;Lowest Level Indicator output
01D0 002A           0663         MOVWF   LED_Mode
                    0664 Task_A
01D1 0B58           0665         GOTO    Task_1          ;Loop Forever
                    0666
                    0667 ;****
                    0668 Do_D_H_E_L      ; Determine the Highest Error Level & Start Task #7
01D2 0C07           0669         MOVLW   H'07'           ;Check top 7 bits
01D3 0027           0670         MOVWF   temp
```


© 1994 Microchip Technology Inc.

```
01D4 0216           0671         MOVF    Old_RB,W        ;Get copy of 7 debounced Sensor Input
01D5 0037           0672         MOVWF   Last_RB
01D6 0377           0673 _0070   RLF     Last_RB         ;Put top bit into C bit
01D7 0603           0674         BTFSC   STATUS,C        ;Check if C bit is set
01D8 0BDE           0675         GOTO    _0072
01D9 02E7           0676         DECFSZ  temp            ;Continue to check lesser bits
01DA 0BD6           0677         GOTO    _0070
01DB 0206           0678 _0071   MOVF    Port_B,W        ;Restore current value of RB
01DC 0037           0679         MOVWF   Last_RB
01DD 0800           0680         RETLW   H'00'
01DE 020A           0681 _0072   MOVF    LED_Mode,W      ;Get current Level Indicator
01DF 0E07           0682         ANDLW   H'07'           ;Get only      "       "
01E0 0037           0683         MOVWF   Last_RB         ;Store into a tempary register
01E1 0207           0684         MOVF    temp,W          ;Check if already at this Level
01E2 0097           0685         SUBWF   Last_RB,W
01E3 0603           0686         BTFSC   STATUS,C
01E4 0BDB           0687         GOTO    _0071
01E5 0C88           0688         MOVLW   H'88'           ;Start to build LED_Mode
01E6 0107           0689         IORWF   temp,W          ;Put new Level Indicator into reg
01E7 002A           0690         MOVWF   LED_Mode        ;Store new LED Mode
01E8 0BDB           0691         GOTO    _0071
                    0692
                    0693 Do_Clear_Regs   ; Clear Registers 7-1Fh
01E9 0C1F           0694         MOVLW   H'1F'           ;First regs to clear
01EA 0024           0695         MOVWF   FSR
01EB 0060           0696 Loop_C  CLRF    INDIR           ;Clear reg
01EC 00E4           0697         DECF    FSR             ;point to next reg to clear
01ED 0CE7           0698         MOVLW   H'E7'           ;Dec temp, jump if not done
01EE 0084           0699         SUBWF   FSR,W
01EF 0603           0700         BTFSC   STATUS,C
01F0 0BEB           0701         GOTO    Loop_C
01F1 0064           0702         CLRF    FSR             ;Lastly clear FSR reg
01F2 0800           0703         RETLW   H'00'
                    0704
                    0705         ORG     H'1FF'          ;RESET to Main
01FF 0B43           0706         GOTO    Main
                    0707
                    0708         END
                    0709
                    0710

Errors   :    0
Warnings :    0
```

© 1994 Microchip Technology Inc.

```
MPASM 01.00.01 Inter.     APP_D.ASM   7-29-1994  11:20:6                PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT

                     0001 ; 'Remote Alarm64' V1.00 Designed by Myriad Development Co. - Jerry Farmer
                     0002 ;       PIC16C64, 4MHz Crystal, WatchDog Timer OFF, MPASM instruction set
                     0003
                     0004         list    p=16C64,t=ON,c=132
                     0005
                     0006         include "PIC16C64.H"
                     0001 ;*************************  PIC16C64 Header    ************************
                     0002 ;Bank 0 - set RP0 = 0
                     0003
0000                 0004 INDF    equ     00h     ;Indirect Address
0001                 0005 TMR0    equ     01h
0002                 0006 PCL     equ     02h
0003                 0007 STATUS  equ     03h
0004                 0008 FSR     equ     04h
0005                 0009 PORTA   equ     05h     ; I/O Port Assignments
0006                 0010 PORTB   equ     06h
0007                 0011 PORTC   equ     07h
0008                 0012 PORTD   equ     08h
0009                 0013 PORTE   equ     09h
000A                 0014 PCLATH  equ     0Ah
000B                 0015 INTCON  equ     0Bh
000C                 0016 PIR1    equ     0Ch
000E                 0017 TMR1L   equ     0Eh
000F                 0018 TMR1H   equ     0Fh
0010                 0019 T1CON   equ     10h
0011                 0020 TMR2    equ     11h
0012                 0021 T2CON   equ     12h
0013                 0022 SSPBUF  equ     13h
0014                 0023 SSPCON  equ     14h
0015                 0024 CCPR1L  equ     15h
0016                 0025 CCPR1H  equ     16h
0017                 0026 CCP1CON equ     17h
                     0027
                     0028 ;BANK 1 - set RP0 = 1
0001                 0029 OPTION  equ     01h
0005                 0030 TRISA   equ     05h
0006                 0031 TRISB   equ     06h
0007                 0032 TRISC   equ     07h
0008                 0033 TRISD   equ     08h
0009                 0034 TRISE   equ     09h
000C                 0035 PIE1    equ     0Ch
000E                 0036 PCON    equ     0Eh
0012                 0037 PR2     equ     12h
```


© 1994 Microchip Technology Inc.

```
0013 SSPADD  equ   13h
0014 SSPSTAT equ   14h
     ;
     ;*****************************************************************************
     ;
0000 C       equ   0       ;STATUS
0001 DC      equ   1       ;STATUS
0002 Z       equ   2       ;STATUS
0003 PD      equ   3       ;STATUS
0004 TO      equ   4       ;STATUS
0005 RP0     equ   5       ;STATUS
0006 RP1     equ   6       ;STATUS
0007 IRP     equ   7       ;STATUS
     ;
0000 RBIF    equ   0       ;INTCON
0001 INTF    equ   1       ;INTCON
0002 RTIF    equ   2       ;INTCON
0003 RBIE    equ   3       ;INTCON
0004 INTE    equ   4       ;INTCON
0005 RTIE    equ   5       ;INTCON
0006 PEIE    equ   6       ;INTCON
0007 GIE     equ   7       ;INTCON
     ;
0000 TMR1IF  equ   0       ;PIR1
0001 TMR2IF  equ   1       ;PIR1
0002 CCP1IF  equ   2       ;PIR1
0003 SSPIF   equ   3       ;PIR1
0007 PSPIF   equ   7       ;PIR1
     ;
0000 TMR1ON  equ   0       ;T1CON
0001 TMR1CS  equ   1       ;T1CON
0002 T1INSYNC equ  2       ;T1CON
0003 T1OSCEN equ   3       ;T1CON
0004 T1CKPS0 equ   4       ;T1CON
0005 T1CKPS1 equ   5       ;T1CON
     ;
0000 T2CKPS0 equ   0       ;T2CON
0001 T2CKPS1 equ   1       ;T2CON
0002 TMR2ON  equ   2       ;T2CON
0003 TOUTPS0 equ   3       ;T2CON
0004 TOUTPS1 equ   4       ;T2CON
0005 TOUTPS2 equ   5       ;T2CON
0006 TOUTPS3 equ   6       ;T2CON
     ;
0000 SSPM0   equ   0       ;SSPCON
0001 SSPM1   equ   1       ;SSPCON
0002 SSPM2   equ   2       ;SSPCON
0003 SSPM3   equ   3       ;SSPCON
```


© 1994 Microchip Technology Inc.

```
0004                 0086 CKP     equ     4               ;SSPCON
0005                 0087 SSPEN   equ     5               ;SSPCON
0006                 0088 SSPOV   equ     6               ;SSPCON
0007                 0089 WCOL    equ     7               ;SSPCON
                     0090
0000                 0091 CCP1M0  equ     0               ;CCP1CON
0001                 0092 CCP1M1  equ     1               ;CCP1CON
0002                 0093 CCP1M2  equ     2               ;CCP1CON
0003                 0094 CCP1M3  equ     3               ;CCP1CON
0004                 0095 CCP1Y   equ     4               ;CCP1CON
0005                 0096 CCP1X   equ     5               ;CCP1CON
                     0097
0000                 0098 TRISE0  equ     0               ;TRISE
0001                 0099 TRISE1  equ     1               ;TRISE
0002                 0100 TRISE2  equ     2               ;TRISE
0004                 0101 PSPMODE equ     4               ;TRISE
0005                 0102 IBOV    equ     5               ;TRISE
0006                 0103 OBF     equ     6               ;TRISE
0007                 0104 IBF     equ     7               ;TRISE
                     0105
0000                 0106 TMR1IE  equ     0               ;PIE1
0001                 0107 TMR2IE  equ     1               ;PIE1
0002                 0108 CCP1IE  equ     2               ;PIE1
0003                 0109 SSPIE   equ     3               ;PIE1
0007                 0110 PSPIE   equ     7               ;PIE1
                     0111
0001                 0112 POR     equ     1               ;PCON
                     0113
0001                 0114 Same    equ     1
0000                 0115 W       equ     0
                     0116
0000                 0117 LSB     equ     0
0007                 0118 MSB     equ     7
                     0119
0001                 0120 TRUE    equ     1
0001                 0121 YES     equ     1
0000                 0122 FALSE   equ     0
0000                 0123 NO      equ     0
                     0124 ;
                     0125 ;***********************************************************************
                     0126
                     0127
                     0128
                     0129
                     0130
                     0006
                     0007
                     0008 ; B Register Definitions
```


© 1994 Microchip Technology Inc.

```
0000                    0009 Serial_IN_1     equ     0               ;Serial Input #1 - 8 bits - INT pin
                        0010                                         ;RB.7 - RB.1 == Input from Sensors
00FF                    0011 RB_TRIS         equ     B'11111111'     ;RB TRIS at INIT State == all input
0000                    0012 RB_MASK         equ     B'00000000'     ;What is High/Low for RB at INIT State
                        0013
                        0014 ; A Register Definitions - Programmable Inputs
0000                    0015 Level_Reset     equ     0               ;PORTA.0 - Reset Level Indicator
0001                    0016 LED             equ     1               ;LED Output - Level/State Indicator
0002                    0017 Serial_Out      equ     2               ;Serial Output - 8 bits + passwords
0003                    0018 PWM_Out         equ     3               ;PWM Output - 8 bits ON, 8 bits OFF
                        0019
00F1                    0020 RA_TRIS         equ     B'11110001'     ;RA TRIS at INIT State
0000                    0021 RA_MASK         equ     B'00000000'     ;What is High/Low for RA at INIT State
                        0022
                        0023 ; Register Files
0020                    0024 temp            equ     20h             ;Tempary holding register - PIC16C54/56
0021                    0025 tmp             equ     21h             ;Tempary reg
0022                    0026 Temp_W          equ     22h             ;Interrupt storage of W
0023                    0027 Temp_Stat       equ     23h             ;Interrupt storage of STATUS
0024                    0028 Temp_FSR        equ     24h             ;Interrupt storage of FSR
0025                    0029 T_B             equ     25h             ;Indicates which Timer(s) are Active = 1
0026                    0030 FLAGS           equ     26h             ;Error Flags
0027                    0031 LED_Mode        equ     27h             ;(0-2)=Mode, 3=LED_B, (4-6)=Seq #, 7=NEW
0028                    0032 OState          equ     28h             ;Serial Out State
0029                    0033 IState1         equ     29h             ;Serial In #1 State
002A                    0034 cc              equ     2Ah             ;256 * TMR0 time
002B                    0035 T_5_M_LO        equ     2Bh             ;5 Min Timer Counter - Low
002C                    0036 T_5_M_HI        equ     2Ch             ;5 Min Timer Counter - High
002D                    0037 T_5_S_CO        equ     2Dh             ;5 Second Timer - lack of Serial Input
002E                    0038 T_20_mS_CO      equ     2Eh             ;20 mS Timer - used for debouncing
002F                    0039 T_PWM_CO        equ     2Fh             ;PWM Counter
0030                    0040 LED_C           equ     30h             ;LED Counter
0031                    0041 xmt_byte        equ     31h             ;Serial xmit byte - destroyed in use
0032                    0042 rcv_byte_1      equ     32h             ;Receive Serial #1 In byte
0033                    0043 RCV_Storage     equ     33h             ;Long term storage of rcv_byte #1
0034                    0044 Old_RB          equ     34h             ;Oldest/Master copy of RB
0035                    0045 Last_RB         equ     35h             ;Last copy of RB
0036                    0046 PWM_ON          equ     36h             ;PWM ON Counter
0037                    0047 PWM_OFF         equ     37h             ;PWM OFF Counter
0038                    0048 PWM_tmp         equ     38h             ;PWM tempary counter
                        0049
                        0050 ; Indicates which Timer(s) are Active = 1 & Flags - T_B
0000                    0051 OState_B        equ     0               ;Serial Out Active Bit
0001                    0052 IState1_B       equ     1               ;Serial IN #1 Active Bit
0002                    0053 T_5_S_B         equ     2               ;5 Second Timer Active Bit
0003                    0054 T_5_M_B         equ     3               ;5 Min Timer Active Bit
0004                    0055 RCV_Got_One_B   equ     4               ;Got a NEW Received byte to send out
0005                    0056 RB_NEW_B        equ     5               ;Indicates a change in RB input
0006                    0057 S_5_S_B         equ     6               ;Serial In 5 secs of inactivity
```

© 1994 Microchip Technology Inc.

```
0007                0058 T_PWM_B         equ     7               ;PWM Activity Bit
                    0059
                    0060 ; Error Flags - FLAGS
0000                0061 FS_Flag_1       equ     0               ;Serial #1 IN had a False Start Error
0001                0062 FE_Flag_1       equ     1               ;Last Serial #1 IN had a Frame Error
0002                0063 RCV_Overflow    equ     2               ;Lost Serial Input Byte - too Slow
0003                0064 RB_Overflow     equ     3               ;Lost RB Input Byte - too Slow
0004                0065 S_5_S_Overflow  equ     4               ;Lost '5S Inactivity' msg - too Slow
0005                0066 Time_Bit        equ     5               ;Indicate 512 uS has passed
                    0067
                    0068 ;Equates for LED Task #7 - LED_Mode
0003                0069 LED_B           equ     3               ;LED is active - LED_Mode.3
0007                0070 LED_NEW_B       equ     7               ;LED has just changed Modes = 1
0008                0071 LED_OFF_MODE    equ     B'00001000'     ;LED OFF
0089                0072 LED_SEQ1_MODE   equ     B'10001001'     ;LED Sequence 1: .2s On, 1s Off
008A                0073 LED_SEQ2_MODE   equ     B'10001010'     ;LED Sequence 2: 3x(.2s), 1s Off
008B                0074 LED_SEQ3_MODE   equ     B'10001011'     ;LED Sequence 3: 5x(.2s), 1s Off
009C                0075 LED_SLOW_MODE   equ     B'10011100'     ;LED Slow Pulsing - .3 Hz
009D                0076 LED_MEDIUM_MODE equ     B'10011101'     ;LED Medium Pulsing - 1 Hz
009E                0077 LED_FAST_MODE   equ     B'10011110'     ;LED Fast Pulsing - 3 Hz
008F                0078 LED_ON_MODE     equ     B'10001111'     ;LED ON Continuously
                    0079
                    0080         ORG     0                       ;Reset Vector
                    0081
0000 28F7           0082         GOTO    Main
                    0083
                    0084         ORG     4                     ;Interrupt Vector
                    0085
0004 29D4           0086         GOTO    Interrupt
                    0087
                    0088 ;******         Task #1 - Asynchronous 9600 Baud Serial Output (LOW=0)
                    0089 Do_OState
0005 0828           0090         MOVF    OState,W              ;Get (0-A) mode #
0006 390F           0091         ANDLW   H'0F'                 ;Get only mode #
0007 0782           0092         ADDWF   PCL                   ;jump to subroutine
0008 2814           0093         GOTO    OStateS               ;Serial Start Bit
0009 2823           0094         GOTO    OState0_7             ;Bit 0
000A 2823           0095         GOTO    OState0_7             ;Bit 1
000B 2823           0096         GOTO    OState0_7             ;Bit 2
000C 2823           0097         GOTO    OState0_7             ;Bit 3
000D 2823           0098         GOTO    OState0_7             ;Bit 4
000E 2823           0099         GOTO    OState0_7             ;Bit 5
000F 2823           0100         GOTO    OState0_7             ;Bit 6
0010 2823           0101         GOTO    OState0_7             ;Bit 7
0011 2829           0102         GOTO    OStateE               ;Serial Stop Bit
0012 2831           0103         GOTO    OStateL               ;Last State
0013 0008           0104         RETURN
                    0105
```


© 1994 Microchip Technology Inc.

```
                        0106 OStateS
0014 3000               0107         MOVLW   H'00'             ;Post & Pre 1=1 & OFF
0015 0092               0108         MOVWF   T2CON
0016 1683               0109         BSF     STATUS,RP0        ;Point to BANK 1
0017 3053               0110         MOVLW   H'68' - H'15'     ;104uS - 9600 Baud & adjust to Latency
0018 0092               0111         MOVWF   PR2
0019 1283               0112         BCF     STATUS,RP0        ;Point to BANK 0
001A 0191               0113         CLRF    TMR2              ;Init to 0
001B 108C               0114         BCF     PIR1,TMR2IF       ;Clear Timer 2 Flag so as to start fresh
001C 1505               0115         BSF     PORTA,Serial_Out  ;Output 'Serial Start Bit' starting Now
001D 1512               0116         BSF     T2CON,TMR2ON      ;Start Timer 2
001E 0AA8               0117         INCF    OState            ;inc to next state BEFORE allowing interrupts
                        0118
001F 1683               0119         BSF     STATUS,RP0        ;Point to BANK 1
0020 148C               0120         BSF     PIE1,TMR2IE       ;Allow for Timer 2 interrupts
0021 1283               0121         BCF     STATUS,RP0        ;Point to BANK 0
0022 0008               0122         RETURN
                        0123
                        0124 OState0_7                         ;Bit 0 - 7
0023 0CB1               0125         RRF     xmt_byte          ;Move bit into C from right most bit
0024 1C03               0126         BTFSS   STATUS,C          ;Output C bit
0025 1105               0127         BCF     PORTA,Serial_Out  ;
0026 1803               0128         BTFSC   STATUS,C          ;
0027 1505               0129         BSF     PORTA,Serial_Out  ;
0028 282A               0130         GOTO    OS_End
                        0131 OStateE
0029 1105               0132         BCF     PORTA,Serial_Out  ;Serial Stop Bit
002A 108C               0133 OS_End  BCF     PIR1,TMR2IF       ;Clear Timer 2 Flag so as to start fresh
002B 1683               0134         BSF     STATUS,RP0        ;Point to BANK 1
002C 306C               0135         MOVLW   H'68' + H'4'      ;104uS - 9600 Baud & adjust to Latency
002D 0092               0136         MOVWF   PR2
002E 1283               0137         BCF     STATUS,RP0        ;Point to BANK 0
002F 0AA8               0138         INCF    OState            ;increment to next state
0030 0008               0139         RETURN
                        0140 OStateL
0031 1683               0141         BSF     STATUS,RP0        ;Point to BANK 1
0032 108C               0142         BCF     PIE1,TMR2IE       ;Do NOT Allow Timer 2 interrupts
0033 1283               0143         BCF     STATUS,RP0        ;Point to BANK 0
                        0144
0034 108C               0145         BCF     PIR1,TMR2IF       ;Clear Timer 2 Flag so as to start fresh
0035 1112               0146         BCF     T2CON,TMR2ON      ;Stop Timer 2
0036 01A8               0147         CLRF    OState            ;Ready to send next byte out
0037 1025               0148         BCF     T_B,OState_B      ;Serial Out not active
0038 0008               0149         RETURN
                        0150
                        0151 ;******          Task #7 - Output Highest Level Indication on LED
                        0152 Do_LED
0039 1BA7               0153         BTFSC   LED_Mode,LED_NEW_B ;Initialize regs if change in modes
```

© 1994 Microchip Technology Inc.

```
003A 284C  0154          GOTO    LED_NEW          ;Inc Counter - Time Unit = 131072 uS
003B 0AB0  0155          INCF    LED_C            ;Get (0-7) mode #
003C 0827  0156          MOVF    LED_Mode,W       ;Get only mode #
003D 3907  0157          ANDLW   H'07'            ;jump to subroutine
003E 0782  0158          ADDWF   PCL
003F 2848  0159          GOTO    LED_OFF          ;LED OFF
0040 2864  0160          GOTO    LED_SEQ1         ;LED Seq 1: 1 short pulse & pause
0041 2867  0161          GOTO    LED_SEQ2         ;LED Seq 2: 2 short pulses & pause
0042 288A  0162          GOTO    LED_SEQ3         ;LED Seq 3: 3 short pulses & pause
0043 2850  0163          GOTO    LED_SLOW         ;LED Slow Pulsing - .3 Hz
0044 285E  0164          GOTO    LED_MEDIUM       ;LED Medium Pulsing - 1 Hz
0045 2861  0165          GOTO    LED_FAST         ;LED Fast Pulsing - 3 Hz
0046 284D  0166          GOTO    LED_ON           ;LED ON Continuously
0047 0008  0167 _0012    RETURN
           0168 ;___
           0169 LED_OFF
0048 1085  0170          BCF     PORTA,LED        ;Turn off LED
0049 11A7  0171          BCF     LED_Mode,LED_B   ;LED must be off
004A 01B0  0172          CLRF    LED_C            ;Reset Counter - LED_C = 0
004B 0008  0173          RETURN
           0174 ;___
           0175 LED_NEW
004C 13A7  0176          BCF     LED_Mode,LED_NEW_B  ;Done initializing
           0177 LED_ON
004D 1485  0178          BSF     PORTA,LED        ;Turn on LED
004E 01B0  0179          CLRF    LED_C            ;Reset Counter - LED_C = 0
004F 0008  0180          RETURN
           0181 ;___
           0182 LED_SLOW
0050 300C  0183          MOVLW   H'0C'            ;.3Hz @ 50% Duty
0051 00A0  0184          MOVWF   temp
0052 0820  0185 LED_S    MOVF    temp,W           ;Check LED_C if time, .3Hz @ 50% Duty
0053 0230  0186          SUBWF   LED_C,W
0054 1D03  0187          BTFSS   STATUS,Z
0055 2847  0188          GOTO    _0012
0056 3010  0189          MOVLW   H'10'
0057 06A7  0190          XORWF   LED_Mode         ;Switch states
0058 1E27  0191          BTFSS   LED_Mode,4       ;Now make LED same state
0059 1085  0192          BCF     PORTA,LED
005A 1A27  0193          BTFSC   LED_Mode,4
005B 1485  0194          BSF     PORTA,LED
005C 01B0  0195          CLRF    LED_C            ;Reset LED_C
005D 0008  0196          RETURN
           0197 ;___
           0198 LED_MEDIUM
005E 3004  0199          MOVLW   H'04'            ;1Hz @ 50% Duty
005F 00A0  0200          MOVWF   temp
0060 2852  0201          GOTO    LED_S            ;Go do it
```


© 1994 Microchip Technology Inc.

```
0202 ;___
0203 LED_FAST
0204                    MOVLW   H'01'           ;3Hz @ 50% Duty
0205                    MOVWF   temp
0206                    GOTO    LED_S           ;Go do it
0207 ;___
0208 LED_SEQ1
0209                    BTFSS   LED_Mode,4      ;Skip if bit is high
0210                    GOTO    ON1             ;Go do it
0211                    GOTO    OFF3            ;Go do it
0212 ;___
0213 LED_SEQ2
0214                    MOVF    LED_Mode,W      ;.2 ON, .2 OFF, .2 ON, 1 OFF
0215                    MOVWF   temp
0216                    MOVLW   H'30'           ;Get sequence # only
0217                    ANDWF   temp
0218                    SWAPF   temp            ;swap nibbles
0219                    MOVF    temp,W          ;get nibble for offset
0220                    ADDWF   PCL             ;Table jump calculation
0221                    GOTO    ON1             ;LED is on, check if time to change
0222                    GOTO    OFF2            ;LED is off, check if time to change
0223                    GOTO    ON1             ;LED is on, check if time to change
0224                    GOTO    OFF3            ;LED is off, check if time to change
0225 ;___
0226 LED_Exit
0227                    MOVLW   H'10'           ;Inc Seq #
0228                    ADDWF   LED_Mode
0229                    CLRF    LED_C           ;Reset LED_C
0230                    RETURN
0231 ON1
0232                    MOVLW   H'02'           ;Check LED_C if time, .2 sec-on
0233                    SUBWF   LED_C,W
0234                    BTFSS   STATUS,Z
0235                    GOTO    _0012           ;Turn off LED
0236                    BCF     PORTA,LED
0237                    GOTO    LED_Exit
0238 OFF2
0239                    MOVLW   H'02'           ;Check LED_C if time, .2 sec-on
0240                    SUBWF   LED_C,W
0241                    BTFSS   STATUS,Z
0242                    GOTO    _0012
0243                    BSF     PORTA,LED       ;Turn on LED
0244                    GOTO    LED_Exit
0245 OFF3
0246                    MOVLW   H'08'           ;Check LED_C if time, 1 sec-off
0247                    SUBWF   LED_C,W
0248                    BTFSS   STATUS,Z
0249                    GOTO    _0012

0061 3001
0062 00A0
0063 2852

0064 1E27
0065 2876
0066 2882

0067 0827
0068 00A0
0069 3030
006A 05A0
006B 0EA0
006C 0820
006D 0782
006E 2876
006F 287C
0070 2876
0071 2882

0072 3010
0073 07A7
0074 01B0
0075 0008

0076 3002
0077 0230
0078 1D03
0079 2847
007A 1085
007B 2872

007C 3002
007D 0230
007E 1D03
007F 2847
0080 1485
0081 2872

0082 3008
0083 0230
0084 1D03
0085 2847
```

5

```
0086 1485                       BSF     PORTA,LED       ;Turn on LED
0087 30F0                       MOVLW   H'F0'
0088 04A7                       IORWF   LED_Mode        ;Cause (Seq# & NEW) to overflow to 0
0089 2872                       GOTO    LED_Exit
                        LED_SEQ3                        ;.2 ON, .2 OFF, .2 ON, .2 OFF, .2 ON, 1 OFF
008A 0827                       MOVF    LED_Mode,W      ;Get LED info
008B 00A0                       MOVWF   temp
008C 3070                       MOVLW   H'70'           ;Get sequence # only
008D 05A0                       ANDWF   temp
008E 0EA0                       SWAPF   temp            ;swap nibbles
008F 0820                       MOVF    temp,W          ;get nibble for offset
0090 0782                       ADDWF   PCL             ;Table jump calculation
0091 2876                       GOTO    ON1             ;LED is on check if time to change
0092 287C                       GOTO    OFF2            ;LED is off check if time to change
0093 2876                       GOTO    ON1             ;LED is on check if time to change
0094 287C                       GOTO    OFF2            ;LED is off check if time to change
0095 2876                       GOTO    ON1             ;LED is on check if time to change
0096 2882                       GOTO    OFF3            ;LED is off check if time to change
                        ;******
                        Do_I1State                      Task #2 - Asynchronous 4800 Baud Serial Input (LOW=0)
0097 0829                       MOVF    IState1,W       ;Get (0-B) mode #
0098 390F                       ANDLW   H'0F'           ;Get only mode #
0099 0782                       ADDWF   PCL             ;jump to subroutine
009A 28A7                       GOTO    I1StateS        ;Serial Start Bit
009B 28B8                       GOTO    I1State2        ;1/2 of Start Bit - see if False Start
009C 28C5                       GOTO    I1State0_7      ;Bit 0
009D 28C5                       GOTO    I1State0_7      ;Bit 1
009E 28C5                       GOTO    I1State0_7      ;Bit 2
009F 28C5                       GOTO    I1State0_7      ;Bit 3
00A0 28C5                       GOTO    I1State0_7      ;Bit 4
00A1 28C5                       GOTO    I1State0_7      ;Bit 5
00A2 28C5                       GOTO    I1State0_7      ;Bit 6
00A3 28C5                       GOTO    I1State0_7      ;Bit 7
00A4 28D4                       GOTO    I1StateE        ;Serial Stop Bit
00A5 28DF                       GOTO    I1StateL        ;Last State - End of Stop Bit
00A6 0008                       RETURN
                        ;***                            Subroutines for Task #2
                        I1StateS                        ;Start Bit - Setup timing variables
00A7 120B                       BCF     INTCON,INTE     ;Disable detecting changes on INT pin
00A8 108B                       BCF     INTCON,INTF     ;Clear Interrupting Flag
00A9 3000                       MOVLW   H'00'           ;Internal Clk, Pre 1=1 & OFF
00AA 0090                       MOVWF   T1CON
00AB 018E                       CLRF    TMR1L           ;Calculate (0 - #) of counts until roll-over
00AC 304A                       MOVLW   H'68' - H'1E'   ;208us/2 = 4800 Baud & adjust to Latency
00AD 028E                       SUBWF   TMR1L
00AE 018F                       CLRF    TMR1H
```


© 1994 Microchip Technology Inc.

```
00AF 038F  0298          DECF    TMR1H           ;H'FF'
00B0 100C  0299          BCF     PIR1,TMR1IF     ;Clear Timer 1 Flag so as to start fresh
00B1 1410  0300          BSF     T1CON,TMR1ON    ;Start Timer 1
00B2 0AA9  0301          INCF    IState1         ;inc to next state BEFORE allowing interrupts
00B3 14A5  0302          BSF     T_B,IState1_B   ;Serial Input Active
           0303
00B4 1683  0304          BSF     STATUS,RP0      ;Point to BANK 1
00B5 140C  0305          BSF     PIE1,TMR1IE     ;Allow for Timer 1 interrupts
00B6 1283  0306          BCF     STATUS,RP0      ;Point to BANK 0
00B7 0008  0307          RETURN
           0308  I1State2                        ;Check if still a Start Bit
00B8 1C06  0309          BTFSS   PORTB,Serial_IN_1  ;False Start Error ?
00B9 28E9  0310          GOTO    FS_Error_1
00BA 1026  0311          BCF     FLAGS,FS_Flag_1 ;Start Bit OK
00BB 1010  0312          BCF     T1CON,TMR1ON    ;Stop Timer 1
00BC 018E  0313          CLRF    TMR1L
00BD 30AA  0314          MOVLW   H'D0' - H'26'   ;208us = 4800 Baud & adjust to Latency
00BE 028E  0315          SUBWF   TMR1L
00BF 018F  0316          CLRF    TMR1H
00C0 038F  0317          DECF    TMR1H           ;H'FF'
00C1 100C  0318          BCF     PIR1,TMR1IF     ;Clear Timer 1 Flag so as to start fresh
00C2 1410  0319          BSF     T1CON,TMR1ON    ;Start Timer 1
00C3 0AA9  0320          INCF    IState1         ;increment to next state
00C4 0008  0321          RETURN
           0322  I1State0_7                      ;Bit 0 - 7
00C5 1C06  0323          BTFSS   PORTB,Serial_IN_1  ;Move Input bit into C
00C6 1003  0324          BCF     STATUS,C
00C7 1806  0325          BTFSC   PORTB,Serial_IN_1
00C8 1403  0326          BSF     STATUS,C
00C9 0CB2  0327          RRF     rcv_byte_1      ;Move C into left most bit
00CA 1010  0328          BCF     T1CON,TMR1ON    ;Stop Timer 1
00CB 018E  0329          CLRF    TMR1L
00CC 30AA  0330          MOVLW   H'D0' - H'26'   ;208us = 4800 Baud & adjust to Latency
00CD 028E  0331          SUBWF   TMR1L
00CE 018F  0332          CLRF    TMR1H
00CF 038F  0333          DECF    TMR1H           ;H'FF'
00D0 100C  0334          BCF     PIR1,TMR1IF     ;Clear Timer 1 Flag so as to start fresh
00D1 1410  0335          BSF     T1CON,TMR1ON    ;Start Timer 1
00D2 0AA9  0336          INCF    IState1         ;increment to next state
00D3 0008  0337          RETURN
           0338  I1StateE                        ;Check if we have a proper Stop Bit
00D4 1806  0339          BTFSC   PORTB,Serial_IN_1  ;Frame Error
00D5 28EC  0340          GOTO    F_Error_1
00D6 10A6  0341          BCF     FLAGS,FE_Flag_1 ;Stop Bit OK
00D7 01AD  0342          CLRF    T_5_S_C0        ;Reset 5 Sec Timer - got a good byte
           0343          ;Process the msg Here !
00D8 0832  0344          MOVF    rcv_byte_1,W    ;Make a copy of just received byte
00D9 00B3  0345          MOVWF   RCV_Storage
```

5

```
00DA 1E25  0346          BTFSS   T_B,RCV_Got_One_B  ;Report Lost data
00DB 1126  0347          BCF     FLAGS,RCV_Overflow
00DC 1A25  0348          BTFSC   T_B,RCV_Got_One_B
00DD 1526  0349          BSF     FLAGS,RCV_Overflow
00DE 1625  0350          BSF     T_B,RCV_Got_One_B  ;We Now have a RB Value to go out
           0351 I1StateL
00DF 1010  0352          BCF     T1CON,TMR1ON    ;Stop Timer 1
00E0 1683  0353          BSF     STATUS,RP0      ;Point to BANK 1
00E1 140C  0354          BSF     PIE1,TMR1IE     ;Allow for Timer 1 interrupts
00E2 1283  0355          BCF     STATUS,RP0      ;Point to BANK 0
00E3 100C  0356          BCF     PIR1,TMR1IF     ;Clear Timer 1 Flag so as to start fresh
           0357
00E4 01A9  0358          CLRF    IState1         ;Ready to receive next byte
00E5 10A5  0359          BCF     T_B,IState1_B   ;Serial In not currently active
           0360
00E6 108B  0361          BCF     INTCON,INTF     ;Clear Interrupting Flag
00E7 160B  0362          BSF     INTCON,INTE     ;Enable detecting changes on INT pin
00E8 0008  0363          RETURN
           0364 FS_Error_1
00E9 10A5  0365          BCF     T_B,IState1_B   ;False Start - Shut Down Checking
00EA 1426  0366          BSF     FLAGS,FS_Flag_1 ;Serial Input NOT Active
00EB 28DF  0367          GOTO    I1StateL        ;False Start Error
           0368 F_Error_1                       ;Start All Over
00EC 14A6  0369          BSF     FLAGS,FE_Flag_1 ;Frame Error - Wait for End of Stop Bit
00ED 0AA9  0370          INCF    IState1         ;Frame Error for this Byte ?
00EE 1010  0371          BCF     T1CON,TMR1ON    ;Increment to next state
00EF 018E  0372          CLRF    TMR1L           ;Stop Timer 1
00F0 304A  0373          MOVLW   H'68' - H'1E'   ;208us/2 = 4800 Baud & adjust to Latency
00F1 028E  0374          SUBWF   TMR1L
00F2 018F  0375          CLRF    TMR1H
00F3 038F  0376          DECF    TMR1H           ;H'FF'
00F4 100C  0377          BCF     PIR1,TMR1IF     ;Clear Timer 1 Flag so as to start fresh
00F5 1410  0378          BSF     T1CON,TMR1ON    ;Start Timer 1
00F6 0008  0379          RETURN
           0380
           0381 ;******          Code Starting point
           0382 Main
00F7 0183  0383          CLRF    STATUS
00F8 0184  0384          CLRF    FSR
00F9 0181  0385          CLRF    TMR0            ;Clear Timer0
00FA 3000  0386          MOVLW   H'00'           ;What is High/Low for RA at RESET State
00FB 0085  0387          MOVWF   PORTA
00FC 0086  0388          MOVWF   PORTB
00FD 0087  0389          MOVWF   PORTC
00FE 0088  0390          MOVWF   PORTD
00FF 0089  0391          MOVWF   PORTE
0100 018A  0392          CLRF    PCLATH
0101 3060  0393          MOVLW   H'60'           ;/GIE,PEIE,RTIE,/INTE,/RBIE,/RTIF,/INTF,/RBIF
```


© 1994 Microchip Technology Inc.

```
0102 008B          0394         MOVWF   INTCON
0103 018C          0395         CLRF    PIR1            ;Timer 2 Flag cleared
0104 018E          0396         CLRF    TMR1L
0105 018F          0397         CLRF    TMR1H
0106 0190          0398         CLRF    T1CON           ;Timer 1 OFF until ready for input
0107 0191          0399         CLRF    TMR2
0108 0192          0400         CLRF    T2CON           ;Timer 2 OFF until have byte to output
0109 0193          0401         CLRF    SSPBUF
010A 0194          0402         CLRF    SSPCON
010B 0195          0403         CLRF    CCPR1L
010C 0196          0404         CLRF    CCPR1H
010D 0197          0405         CLRF    CCP1CON
                   0406
010E 1683          0407         BSF     STATUS,RP0      ;Point to BANK 1
010F 3040          0408         MOVLW   H'40'           ;TMR0/2 & Interrupt on Rising edge of INT
0110 0081          0409         MOVWF   OPTION          ;Load OPTION reg
0111 30F1          0410         MOVLW   RA_TRIS
0112 0085          0411         MOVWF   TRISA
0113 30FF          0412         MOVLW   H'FF'           ;RB TRIS at RESET State
0114 0086          0413         MOVWF   TRISB
0115 0087          0414         MOVWF   TRISC
0116 0088          0415         MOVWF   TRISD
0117 3007          0416         MOVLW   H'07'           ;PSPMODE=0
0118 0089          0417         MOVWF   TRISE
0119 018C          0418         CLRF    PIE1            ;Timer 2 Interrupt disabled
011A 148E          0419         BSF     PCON,POR
011B 0192          0420         CLRF    PR2
011C 0193          0421         CLRF    SSPADD
011D 0194          0422         CLRF    SSPSTAT
                   0423
011E 1283          0424         BCF     STATUS,RP0      ;Point to BANK 0
011F 21BF          0425         CALL    Clear_Regs      ;Clear Registers 20-7F, A0-C0 Memory Pages
                   0426
                   0427 ;Initialize Tasks
                   0428                                 ;Task #1 waits for byte to output
                   0429                                 ;Task #2 waits for Serial IN Start Bit
0120 3031          0430         MOVLW   H'31'           ;Task #3 is initialized for square pulses
0121 00B6          0431         MOVWF   PWM_ON          ;   "     25 mS ON
0122 3031          0432         MOVLW   H'31'           ;   "             Period = 50 mS, DS= 50%
0123 00B7          0433         MOVWF   PWM_OFF         ;   "     25 mS OFF
0124 01AF          0434         CLRF    T_PWM_CO        ;   "
0125 0836          0435         MOVF    PWM_ON,W        ;mov PWM_tmp,PWM_ON
0126 00B8          0436         MOVWF   PWM_tmp         ; "
0127 1585          0437         BSF     PORTA,PWM_Out   ;Start Outputing ON
0128 17A5          0438         BSF     T_B,T_PWM_B     ;   "
                   0439                                 ;Task #4 runs when Task 1 is Not
                   0440                                 ;Task #5 is always running
0129 0806          0441         MOVF    PORTB,W         ;Task #6 is Initialized here
```

© 1994 Microchip Technology Inc.

```
012A 00B4                0442          MOVWF   Old_RB
012B 0834                0443          MOVF    Old_RB,W        ;Make all the same initial value
012C 00B5                0444          MOVWF   Last_RB
012D 16A5                0445          BSF     T_B,RB_NEW_B    ;Tell Task #4: RB byte ready to output
012E 3008                0446          MOVLW   LED_OFF_MODE
012F 00A7                0447          MOVWF   LED_Mode        ;Task #7 is Started
0130 1525                0448          BSF     T_B,T_5_S_B     ;Task #8 is Started here
0131 15A5                0449          BSF     T_B,T_5_M_B     ;Task #9 is Started here
                         0450
0132 178B                0451          BSF     INTCON,GIE      ;Enable Global Interrupts
                         0452
                         0453  ; Handle Task & Timer activities - Main Loop done in background
                         0454  Inc_Time      ;Heart Beat - Time unit = 512 uS for Tasks #5 & #6
0133 1EA6                0455          BTFSS   FLAGS,Time_Bit  ;Idle Task - wait until 512 uS has gone by
0134 2933                0456          GOTO    Inc_Time
0135 12A6                0457          BCF     FLAGS,Time_Bit  ;Reset for next indicator from TMR0 Interrupt
                         0458
                         0459  Task_3          ;Task #3 - PWM, Period = (PWM_ON + PWM_OFF) * 512uS
0136 1FA5                0460          BTFSS   T_B,T_PWM_B     ;if NOT outputing now then skip call
0137 2947                0461          GOTO    Task_4
0138 0AAF                0462          INCF    T_PWM_CO        ;Inc count of time
0139 0838                0463          MOVF    PWM_tmp,W       ;cjne T_PWM_CO,PWM_tmp,Task_4
013A 022F                0464          SUBWF   T_PWM_CO,W      ; "
013B 1D03                0465          BTFSS   STATUS,Z        ; "
013C 2947                0466          GOTO    Task_4          ; "
013D 01AF                0467          CLRF    T_PWM_CO        ;Reset timer
013E 1985                0468          BTFSC   PORTA,PWM_Out
013F 2944                0469          GOTO    T3_1
0140 1585                0470          BSF     PORTA,PWM_Out   ;Change Output State
0141 0836                0471          MOVF    PWM_ON,W        ;mov PWM_tmp,PWM_ON
0142 00B8                0472          MOVWF   PWM_tmp         ; "
0143 2947                0473          GOTO    Task_4
                         0474  T3_1
0144 1185                0475          BCF     PORTA,PWM_Out   ;Change Output State
0145 0837                0476          MOVF    PWM_OFF,W       ;mov PWM_tmp,PWM_OFF
0146 00B8                0477          MOVWF   PWM_tmp         ; "
                         0478
                         0479  Task_4          ;Task #4 - Finds next Buffered Byte to Send Out through Task 1
0147 1825                0480          BTFSC   T_B,OState_B    ;if outputing now then skip call
0148 295E                0481          GOTO    Task_5
0149 1E25                0482          BTFSS   T_B,RCV_Got_One_B  ;Got a NEW Received byte to send
014A 2950                0483          GOTO    _0057
014B 0833                0484          MOVF    RCV_Storage,W   ;Send just received byte
014C 00B1                0485          MOVWF   xmt_byte
014D 1225                0486          BCF     T_B,RCV_Got_One_B  ;Clear need to send old byte
014E 1425                0487          BSF     T_B,OState_B    ;Start Task #1 & Lock Out Others
014F 295D                0488          GOTO    T4_S
0150 1EA5                0489  _0057   BTFSS   T_B,RB_NEW_B    ;Indicates a change in RB input
```


© 1994 Microchip Technology Inc.

```
0151 2957           0490          GOTO    _0058
0152 0834           0491          MOVF    Old_RB,W        ;Send New RB value
0153 00B1           0492          MOVWF   xmt_byte
0154 12A5           0493          BCF     T_B,RB_NEW_B    ;Clear need to send out newest value
0155 1425           0494          BSF     T_B,OState_B    ;Start Task #1 & Lock Out Others
0156 295D           0495          GOTO    T4_S
0157 1F25           0496 _0058    BTFSS   T_B,S_5_S_B     ;Serial In 5 secs of inactivity
0158 295E           0497          GOTO    Task_5
0159 30FF           0498          MOVLW   H'FF'           ;Tell of inactivity of Serial In
015A 00B1           0499          MOVWF   xmt_byte
015B 1325           0500          BCF     T_B,S_5_S_B     ;Clear need to send msg
015C 1425           0501          BSF     T_B,OState_B    ;Start Task #1 & Lock Out Others
                    0502 T4_S                             ;Start Task #1
015D 2005           0503          CALL    Do_OState
                    0504
                    0505 Task_5           ;Task #5 - Monitor Level Reset Input Line - Always Running !
015E 1805           0506          BTFSC   PORTA,Level_Reset
015F 2962           0507          GOTO    Task_6
0160 3008           0508          MOVLW   LED_OFF_MODE    ;Lowest Level Indicator output
0161 00A7           0509          MOVWF   LED_Mode
                    0510
                    0511 Task_6           ;Task #6 - Debounce 8 bit Input Sensors - Runs every 20 mS
0162 0AAE           0512          INCF    T_20_mS_CO      ;Inc Counter - Time Unit = 512 uS
0163 3027           0513          MOVLW   H'27'           ;Used to debounce the input
0164 022E           0514          SUBWF   T_20_mS_CO,W
0165 1D03           0515          BTFSS   STATUS,Z
0166 297C           0516          GOTO    _0065
0167 01AE           0517          CLRF    T_20_mS_CO      ;Reset T_20_mS_CO to start over again
0168 0806           0518          MOVF    PORTB,W         ;Last copy of RB same as Current ?
0169 0235           0519          SUBWF   Last_RB,W
016A 1903           0520          BTFSC   STATUS,Z
016B 296F           0521          GOTO    _0062
016C 0806           0522          MOVF    PORTB,W         ;Store Current RB - diff from Last
016D 00B5           0523          MOVWF   Last_RB
016E 2971           0524          GOTO    _0063
016F 0835           0525 _0062    MOVF    Last_RB,W       ;New Old RB <- same value over 20 mS
0170 00B4           0526          MOVWF   Old_RB
0171 08B4           0527 _0063    MOVF    Old_RB          ;See if RB is now 0
0172 1903           0528          BTFSC   STATUS,Z        ;RB == 0 ? then keep timer running
0173 2976           0529          GOTO    _0064
0174 01AB           0530          CLRF    T_5_M_LO        ;Reset 5 Min Timer
0175 01AC           0531          CLRF    T_5_M_HI        ;  still not zero yet
0176 21A8           0532 _0064    CALL    D_H_E_L         ;Determine the Highest Error Level
0177 1EA5           0533          BTFSS   T_B,RB_NEW_B    ;Check for Lost Data Error
0178 11A6           0534          BCF     FLAGS,RB_Overflow
0179 1AA5           0535          BTFSC   T_B,RB_NEW_B
017A 15A6           0536          BSF     FLAGS,RB_Overflow
017B 16A5           0537          BSF     T_B,RB_NEW_B    ;Every 20 mS send Old_RB out
```

© 1994 Microchip Technology Inc.

```
                        0538
                        0539                    ;Heart Beat - Time unit = 131072 uS for Tasks #7, #8 & #9
                        0540 _0065
017C 1683               0541            BSF     STATUS,RP0      ;Point to BANK 1
017D 30F1               0542            MOVLW   RA_TRIS         ;RA TRIS - refresh
017E 0085               0543            MOVWF   TRISA
017F 30FF               0544            MOVLW   H'FF'           ;RB TRIS - refresh
0180 0086               0545            MOVWF   TRISB
0181 1283               0546            BCF     STATUS,RP0      ;Point to BANK 0
0182 0BAA               0547            DECFSZ  cc              ;Step-up time units * 256
0183 2933               0548            GOTO    Inc_Time
                        0549
                        0550 Task_7             ;Task 7 - Output Highest Level Indication on LED
0184 1DA7               0551            BTFSS   LED_Mode,LED_B  ;Is LED active ?
0185 2987               0552            GOTO    Task_8
0186 2039               0553            CALL    Do_LED          ;Handle LED timing
                        0554
                        0555 Task_8             ;Task #8 - 5 Second Serial Input Lack of Activity Timer
0187 1D25               0556            BTFSS   T_B,T_5_S_B     ;5 Sec Timer Active ?
0188 2996               0557            GOTO    Task_9
0189 0AAD               0558            INCF    T_5_S_CO        ;Inc Counter - Time Unit = 131072 uS
018A 3026               0559            MOVLW   H'26'           ;Check T_5_S_CO if time
018B 022D               0560            SUBWF   T_5_S_CO,W
018C 1D03               0561            BTFSS   STATUS,Z
018D 2996               0562            GOTO    Task_9
018E 01AD               0563            CLRF    T_5_S_CO        ;Reset T_5_S_CO
018F 308F               0564            MOVLW   LED_ON_MODE     ;Highest Level Indicator output
0190 00A7               0565            MOVWF   LED_Mode
0191 1F25               0566            BTFSS   T_B,S_5_S_B     ;Check if Lost Data Error
0192 1226               0567            BCF     FLAGS,S_5_S_Overflow
0193 1B25               0568            BTFSC   T_B,S_5_S_B
0194 1626               0569            BSF     FLAGS,S_5_S_Overflow
0195 1725               0570            BSF     T_B,S_5_S_B     ;Send notice of 5 seconds of inaction
                        0571
                        0572 Task_9             ;Task #9 - 5 Min. Lack of Severe Error from Sensors Reset Timer
0196 1DA5               0573            BTFSS   T_B,T_5_M_B     ;5 Min Timer Active ?
0197 29A7               0574            GOTO    Task_A
0198 0AAB               0575            INCF    T_5_M_LO        ;Inc LO Counter; Time Unit = 131072 uS
0199 1903               0576            BTFSC   STATUS,Z        ;See if carry needs to be passed on ?
019A 0AAC               0577            INCF    T_5_M_HI        ;Inc HI Counter; Time Unit = 131072 uS
019B 3008               0578            MOVLW   H'08'           ;#2288<  Check T_5_M_HI if time
019C 022C               0579            SUBWF   T_5_M_HI,W
019D 1D03               0580            BTFSS   STATUS,Z
019E 29A7               0581            GOTO    Task_A
019F 30F0               0582            MOVLW   H'F0'           ;#2288>  Check T_5_M_LO if time
01A0 022B               0583            SUBWF   T_5_M_LO,W
01A1 1D03               0584            BTFSS   STATUS,Z
01A2 29A7               0585            GOTO    Task_A
```


© 1994 Microchip Technology Inc.

```
01A3 01AB  0586          CLRF    T_5_M_LO        ;Reset T_5_M_LO
01A4 01AC  0587          CLRF    T_5_M_HI        ;Reset T_5_M_HI
01A5 3008  0588          MOVLW   LED_OFF_MODE    ;Lowest Level Indicator output
01A6 00A7  0589          MOVWF   LED_Mode
           0590 Task_A
01A7 2933  0591          GOTO    Inc_Time        ;Loop Forever
           0592
           0593 ;****    Determine the Highest Error Level & Start Task #7
           0594 D_H_E_L
01A8 3007  0595          MOVLW   H'07'           ;Check top 7 bits
01A9 00A0  0596          MOVWF   temp
01AA 0834  0597          MOVF    Old_RB,W        ;Get copy of 7 debounced Sensor Input
01AB 00A1  0598          MOVWF   tmp
01AC 0DA1  0599 _0070    RLF     tmp             ;Put top bit into C bit
01AD 1803  0600          BTFSC   STATUS,C        ;Check if C bit is set
01AE 29B4  0601          GOTO    _0072
01AF 0BA0  0602          DECFSZ  temp            ;Continue to check lesser bits
01B0 29AC  0603          GOTO    _0070
01B1 0806  0604 _0071    MOVF    PORTB,W         ;Restore current value of RB
01B2 00A1  0605          MOVWF   tmp
01B3 0008  0606          RETURN
01B4 0827  0607 _0072    MOVF    LED_Mode,W      ;Get current Level Indicator
01B5 3907  0608          ANDLW   H'07'           ;Get only    "
01B6 00A1  0609          MOVWF   tmp             ;Store into a tempary register
01B7 0820  0610          MOVF    temp,W          ;Check if already at this Level
01B8 0221  0611          SUBWF   tmp,W
01B9 1803  0612          BTFSC   STATUS,C
01BA 29B1  0613          GOTO    _0071
01BB 3088  0614          MOVLW   H'88'           ;Start to build LED_Mode
01BC 0420  0615          IORWF   temp,W          ;Put new Level Indicator into reg
01BD 00A7  0616          MOVWF   LED_Mode        ;Store new LED Mode
01BE 29B1  0617          GOTO    _0071
           0618
           0619 ;******  Clear Registers 20-7Fh, A0-C0
           0620 Clear_Regs
01BF 307F  0621          MOVLW   H'7F'           ;First regs to clear in Bank 0
01C0 0084  0622          MOVWF   FSR
01C1 0180  0623 Loop_C1  CLRF    INDF            ;Clear reg
01C2 0384  0624          DECF    FSR             ;point to next reg to clear
01C3 3020  0625          MOVLW   H'20'           ;Dec temp, jump if not done
01C4 0204  0626          SUBWF   FSR,W
01C5 1803  0627          BTFSC   STATUS,C
01C6 29C1  0628          GOTO    Loop_C1
           0629
01C7 30C0  0630          MOVLW   H'C0'           ;First regs to clear in Bank 1
01C8 0084  0631          MOVWF   FSR
01C9 0180  0632 Loop_C2  CLRF    INDF            ;Clear reg
01CA 0384  0633          DECF    FSR             ;point to next reg to clear
```


© 1994 Microchip Technology Inc.

```
01CB 30A0           0634         MOVLW   H'A0'           ;Dec temp, jump if not done
01CC 0204           0635         SUBWF   FSR,W
01CD 1803           0636         BTFSC   STATUS,C
01CE 29C9           0637         GOTO    Loop_C2
01CF 0184           0638         CLRF    FSR             ;Lastly clear FSR reg
01D0 0008           0639         RETURN
                    0640
                    0641 ;******         TMR0 IRS - Set Time_Bit for background tasks
                    0642 Do_Inc_Time
01D1 16A6           0643         BSF     FLAGS,Time_Bit  ;Tell background tasks of overflow
01D2 110B           0644         BCF     INTCON,RTIF     ;Clear for next overflow
01D3 0008           0645         RETURN
                    0646
                    0647 ;******         Handle Interrupts Here
                    0648 Interrupt
01D4 00A2           0649 PUSH:   MOVWF   Temp_W
01D5 0E03           0650         SWAPF   STATUS,W
01D6 00A3           0651         MOVWF   Temp_Stat
01D7 0804           0652         MOVF    FSR,W
01D8 00A4           0653         MOVWF   Temp_FSR
01D9 1283           0654         BCF     STATUS,RP0      ;Point to BANK 0 - Very IMPORTANT !!!!!
                    0655
01DA 188C           0656         BTFSC   PIR1,TMR2IF
01DB 2005           0657         CALL    Do_OState       ;Go Do Task #1 - all states
                    0658
01DC 18A5           0659         BTFSC   T_B,IState1_B   ;INTF will set even if INTE is cleared
01DD 29E0           0660         GOTO    I1
01DE 188B           0661         BTFSC   INTCON,INTF
01DF 2097           0662         CALL    Do_I1State      ;Go Do Task #2 - 0 state only
                    0663 I1:
01E0 180C           0664         BTFSC   PIR1,TMR1IF
01E1 2097           0665         CALL    Do_I1State      ;Go Do Task #2 - 1-B states
                    0666
01E2 190B           0667         BTFSC   INTCON,RTIF
01E3 21D1           0668         CALL    Do_Inc_Time     ;Go Inc Time_Bit every 512uS
                    0669
01E4 0824           0670 POP:    MOVF    Temp_FSR,W
01E5 0084           0671         MOVWF   FSR
01E6 0E23           0672         SWAPF   Temp_Stat,W
01E7 0083           0673         MOVWF   STATUS
01E8 0EA2           0674         SWAPF   Temp_W
01E9 0E22           0675         SWAPF   Temp_W,W
01EA 0009           0676         RETFIE                  ;Return from Interrupt
                    0677
                    0678         END
                    0679
                    0680

Errors   :     0
Warnings :     0
```


© 1994 Microchip Technology Inc.

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602 786-7200 Fax: 602 786-7277
Technical Support: 602 786-7627
Web: http://www.mchip.com/microhip

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

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508 480-9990 Fax: 508 480-8575

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 708 285-0071 Fax: 708 285-0075

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 214 991-7177 Fax: 214 991-8588

Dayton
Microchip Technology Inc.
35 Rockridge Road
Englewood, OH 45322
Tel: 513 832-2543 Fax: 513 832-2841

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 455
Irvine, CA 92715
Tel: 714 263-1888 Fax: 714 263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516 273-5305 Fax: 516 273-5335

AMERICAS (continued)

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

ASIA/PACIFIC

Hong Kong
Microchip Technology
Unit No. 3002-3004, Tower 1
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T. Hong Kong
Tel: 852 2 401 1200 Fax: 852 2 401 3431

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

Singapore
Microchip Technology
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65 334 8870 Fax: 65 334 8850

Taiwan
Microchip Technology
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2 717 7175 Fax: 886 2 545 0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44 0 1628 851077 Fax: 44 0 1628 850259

France
Arizona Microchip Technology SARL
2 Rue du Buisson aux Fraises
91300 Massy - France
Tel: 33 1 69 53 63 20 Fax: 33 1 69 30 90 79

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

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Pegaso Ingresso No. 2
Via Paracelso 23, 20041
Agrate Brianza (MI) Italy
Tel: 39 039 689 9939 Fax: 39 039 689 9883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shin Yokohama
Kohoku-Ku, Yokohama
Kanagawa 222 Japan
Tel: 81 45 471 6166 Fax: 81 45 471 6122

9/22/95

All rights reserved. © 1995, Microchip Technology Incorporated, USA.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. 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. The Microchip logo and name are registered trademarks of Microchip Technology Inc. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.