```
InitMaster
       Call MotorDriveInit()
       Call SpecialCommInit()
       Call SCI_Init()
       Call InitTimers()
       Initialize Servo Library with SERVOSTRING (#define)
       Initialize Analog Pins with ANALOGSTRING (#define)
       Call ES_PostToService  //Post Transition Event

PostMaster
       Call ES_PostToService

RunMaster
       if ThisEvent.EventType is ES_TIMEOUT and ThisEvent.EventParam is SPECIAL_TIMER
              if SpecialCommandState is 0
                     set SpecialCommandState to 1
                     set COMMAND_PIN high // tell PIC to update Special on lamps
                     reset and start ES_Timer (SPECIAL_TIMER) to 100 ms

              else if SpecialCommandState is 1
                     set SpecialCommandState to 0
                     set COMMAND_PIN low // tell PIC to update Special on batteries
                     reset and start ES_Timer (SPECIAL_TIMER) to 1000 ms


       else if ThisEvent.EventType is ES_TIMEOUT and ThisEvent.EventParam is 3
              // timeout for disconnecting if haven't received new messages
              if connected is true
                     set ConnectedSourceMSB and LSB to 0
                     call SetMotorDriveDuty(128, MOTOR_LA) // turn left motor off
                     call SetMotorDriveDuty(128, MOTOR_RA) // turn right motor off
                     set connected to 0
                     set active to False
                     set current transmit state to TX_START

       else if ThisEvent.EventType is ES_LAMP1
              set Lamp1TX to ThisEvent.EventParam (1 or 0 for on or off)

       else if ThisEvent.EventType is ES_LAMP2
              set Lamp2TX to ThisEvent.EventParam

       else if ThisEvent.EventType is ES_LAMP3
              set Lamp3TX to ThisEvent.EventParam

       else if ThisEvent.EventType == ES_CHARGE_STATE
              ChargeTX = ThisEvent.EventParam (1 or 0 for charging or discharging

  return ReturnEvent;

/*****************************************************************************
 INIT FUNCTIONS
 *****************************************************************************/


SpecialCommInit()
       set lamp output state registers
       set lamp input state registers
       set charge status input registers

       //set up input capture system
       set TSCR1 to _S12_TEN // turn timer subsystem on, doesn't stop timer in freeze
       set TSCR2 to (_S12_PR2 |_S12_PR0)  // set to divde by 128
        set TIOS &= ~_S12_IOS7 // set cap/comp 3 to input capture  (0 = input capture)
```

```
      set TCTL3 |= (_S12_EDG7A | _S12_EDG7B) // capture rising and falling edges
      set TFLG1 = _S12_C7F // clear IC7 flag
      set TIE |= _S12_C7I // enable IC7 interrupt


SCI_Init()
       SCIBDH = 0;
       SCIBDL = 156; // set baud rate to 9600
       // control register 1
       set SCICR2 = ( _S12_RIE | _S12_TE |_S12_RE) // transmit enable, receive enable

InitTimers()
       set and start special timer to 200

}


/*************************************************************************
 COMMUNICATION FUNCTIONS
 *************************************************************************/

sendByte(char msg )
       if ((SCISR1 & _S12_TDRE) != 0)// if ready to send
              set SCIDRL to msg
              return TRUE
       else
              return FALSE

sendSM
       switch CurrentTXState
              case TX_START:
                     call sendByte(0x7E)
                            if return is true,  CurrentTXState = TX_MSBLength
                     checksum = 0

              case TX_MSBLength:
                     call sendByte(0x00)
                            if return is true,  CurrentTXState TX_LSBLength

              case TX_LSBLength:
                     call sendByte(0x0A)
                            if return is true,  CurrentTXState TX_API

              case TX_API:
                     call sendByte(0x01)
                            if return is true
                                   CurrentTXState TX_FRAME
                                   checksum += 0x01

              case TX_FRAME:
                     call sendByte 0x01
                            if return is true
                                   CurrentTXState TX_MSBDestination
                                   checksum += FrameID;
                            if FrameID is greater than  50, set FrameID = 0x01

              case TX_MSBDestination:
                     call sendByte ConnectedSourceMSB
                            if return is true
                            CurrentTXState is TX_LSBDestination
                            chesksum += ConnectedSourceMSB

              case TX_LSBDestination:
```

```
       call sendByte ConnectedSourceLSB
              if return is true
              CurrentTXState is TX_OPTIONS
              chesksum += ConnectedSourceLSB


case TX_OPTIONS:
       call sendByte TXOptions
              if return is true
              CurrentTXState is TX_STATUS
              chesksum += TXOptions

case TX_STATUS:
   if MessageType is DisconnectedMessage and connect is 0
       ConnectedSourceMSB = 0x00
       ConnectedSourceLSB = 0x00
       byteToSend = 0x05

   else if MessageType is ConnectMessage and connected is 1
       byteToSend = 0x04

   else byteToSend is 0x03
       if sendByte(byteToSend) returns true
              CurrentTXState is TX_BATT
              checksum += byteToSend
 case TX_BATT:
       take average of last 10 BatteryPeriods
       byteToSend = Average
       checksum += Average
       call sendByte byteToSend
              if return is true
                     CurrentTXState is TX_CHARGE
                     chesksum += byteToSend


case TX_CHARGE:
       set byteToSend = ChargeTX

       call sendByte byteToSend
              if return is true
                     CurrentTXState is TX_LAMP
                     checksum += byteToSend

case TX_LAMP:
       set byteToSend = (Lamp1TX |Lamp2TX |Lamp3TX)
       if(sendByte(byteToSend))
              CurrentTXState = TX_CAMERA
              checksum += byteToSend

case TX_CAMERA:
       set byteToSend = LastCamCom
              if(sendByte(byteToSend))
                     CurrentTXState = TX_Checksum
                     checksum += byteToSend

case TX_Checksum:

       if(sendByte(0xFF - (checksum & 0xFF)))
              CurrentTXState = TX_START
              SCICR2 &= ~_S12_TIE
              sending = FALSE
```

```
receiveSM(void )
	if SCISR1 & _S12_RDRF isn't 0 // if interrupt enabled and ready to recieve
		input = SCIDRL
		input &= 0xFF
	if input & 0xFF == HeaderByte and active == FALSE
		active = TRUE
		i = 2
		receiving = TRUE
		receive[1] = HeaderByte
	else if (active == TRUE)
		receive[i] = input// i is the byte number
		switch(i)

			case 2:
			   MSBLength = receive[i] & 0xFF

			case 3:
				LSBLength = receive[i] & 0xFF
				Length = (int)(LSBLength);

				if (Length < 1 || Length > 30)
					Length = 3

			case 4:
				APIIdent = receive[i] & 0xFF

			case 5:
				correctsource = 1
				if(APIIdent == 0x81)
					SourceMSB = receive[i] & 0xFF;
					if (connected ==1)

						if(SourceMSB != ConnectedSourceMSB)
							correctsource =0;
							active = FALSE;

			case 6:
				if(APIIdent == 0x81)


				SourceLSB = receive[i] & 0xFF
				Source = (int)(SourceLSB + (SourceMSB << 8))

				// if connected to a POD, make sure this is the correct POD
				if (connected ==1)                        {
					if(SourceLSB != ConnectedSourceLSB)
						correctsource =0
						active = FALSE

				else if (APIIdent == 0x89)
			case 7:
				if(APIIdent == 0x81), RSSI = receive[i]

			case 8:
				if(APIIdent == 0x81)
					Options = receive[i]

			case RXFirst:

				if (APIIdent == 0x81)  // figure out message type
					if(receive[i] == 0x00), MessageType = ConnectMessage
				else if(receive[i] == 0x01)
```

```
                                MessageType = DisconnectMessage;
                                       if (connected == 0) correctsource = 0
                                       if (correctsource == 1 && connected == 1)
                                              connected = 0;
                                              SetMotorDriveDuty(128, MOTOR_RA)
                                              SetMotorDriveDuty(128, MOTOR_LA)

                                else if(receive[i] == 0x02)
                                       MessageType = CommandMessage;
                                        if (connected == 0), correctsource = 0

                                if(correctsource == 0 and connected == 1) MessageType =
WrongSourceMessage                              }

                        case RXSecond:

                        if (APIIdent == 0x81)
                               left = (int)(receive[i] & 0xFF);
                               if(MessageType is CommandMessage && connected == 1 and
correctsource == 1)
                                       if cameraForward is 1
                                              SetMotorDriveDuty(left, MOTOR_LA)
                                       else
                                              SetMotorDriveDuty(255-left, MOTOR_RA)
                               else if(MessageType == ConnectMessage && connected == 0)
                                       // check if roamer number is correct
                                       if(receive[i] is 1
                                              if PTAD & BIT1HI isn't equal to 0
                                                     ConnectedSourceLSB = SourceLSB
                                                     connected = 1
                                       if(receive[i] is 2
                                              if PTAD & BIT2HI isn't equal to 0
                                                     ConnectedSourceLSB = SourceLSB
                                                     connected = 1

                                       if(receive[i] is 3
                                              if PTAD & BIT3HI isn't equal to 0
                                                     ConnectedSourceLSB = SourceLSB
                                                     connected = 1




                        case RXRight:

                               if (APIIdent == 0x81)
                                      right = receive[i]
                           if(MessageType == CommandMessage && connected ==1 &&
correctsource ==1)
                              // right motor command

                                       if cameraForward is 1
                                              SetMotorDriveDuty(right, MOTOR_RA);
                                       else
                                              SetMotorDriveDuty(255-right, MOTOR_LA);

                        case RXGripper:
                        if APIIdent == 0x81
                               if(MessageType ==CommandMessage && connected ==1 &&
correctsource ==1)
                                       // gripper command
                                       call Servo_SetAngle(GrabberServo,receive[i])
```

```
            case RXCamera:
                  if APIIdent is 0x81
                     if MessageType ==CommandMessage && connected ==1 &&
correctsource ==1
                        call Servo_SetAngle(SpecialServo,receive[i] & 0xFF)
                        LastCamCom = receive[i]

                        if LastCamCom & BIT7HI isn't 0
                           cameraForward = TRUE;
                        else
                           cameraForward = FALSE;

            case RXDigital:
                if APIIdent is 0x81
                   if MessageType is CommandMessage && connected ==1 && correctsource
==1
                      SpecialLedRX = receive[i]

                          if SpecialLedRX & BIT0HI isn't 0, turn led 1 on
                          else, turn led 1 off

                          if SpecialLedRX & BIT0HI isn't 0, turn led 2 on
                          else, turn led 2 off

                          if SpecialLedRX & BIT0HI isn't 0, turn led 3 on
                          else, turn led 3 off

// Check checksum
         if i > Length + 3
             checksumindex = 4 // overflowed
              checksum = 0

         for (checksumindex; checksumindex < i; checksumindex++)
                  checksum += receive[checksumindex]

              if ((APIIdent == 0x81) && correctsource == 1)

                     set sending equal to TRUE
                     SCICR2 |= _S12_TIE // enable transmit
                     start ES_SetTimer (timer 3) to 1000 ms // disconnect timer

         i++

 /*************************************************************************
 intertupt functions
 *************************************************************************/

interrupt _Vec_TIMER7 BatLifeMeasure {

   set CurrentTime = TC7
   set TFLG1 = _S12_C7F // clear IC7 Flag
   set PinState = PTT & BIT7HI  // read current pin
   if PinState is zero // it was a falling edge
       BatteryPeriod[i] equals (CurrentTime - LastRisingEdge)
              if i <9, i++
              else i = 0
   LastRisingEdge = CurrentTime


interrupt _Vec_SCI SCI_Interrupt{ // received new message

   if SCISR1 & _S12_RDRF isn't equal to 0 // ready to receive
```

```
      call receiveSM()

   else if SCISR1 & _S12_TDRE isn't equal to 0 and sending is TRUE// sending, don't
receive
       call sendSM()
```