Graphics Microsystems Inc.

PCU 100
Project

Abstract:

This document contains all of the information regarding the operation of the PCU 100 (**P**ress **C**ontrol **U**nit).

Revision History

Author	Revision	Date	Remarks
Paul L Calinawan	00.01	November 26, 1996	Initial release
Paul L Calinawan	00.02	January 7, 1997	Made revisions based on the SPU Functional Specification Review
Date Printed : 12/18/2017			

Author	Revision	Date	Remarks
Paul L Calinawan	00.03	March 14, 1997	First Distribution
Paul L Calinawan	00.04	April 28, 1997	Added state machine documentation and made updates to existing ones

Table of Contents

1. General Description

1.1 Product Overview

The PCU is designed to be for multiple applications. It can serve any of the following purpose:

1. Analog Feedback
2. Motor Control using Analog Feedback (w limit switch detection)
3. 3 Digital Input Monitor
4. 3 Digital Output Control
5. Sheet Counter
6. All functions combined

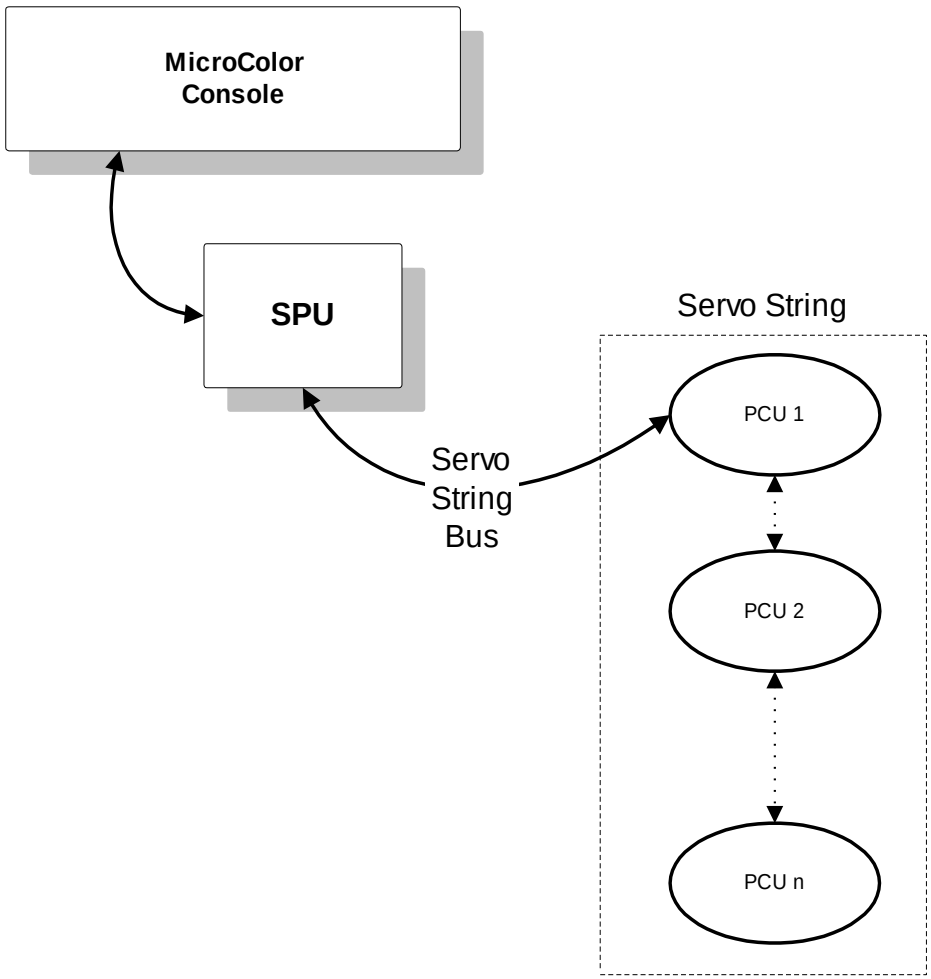
Just like the "Old Servo Units"[1], the PCU connects to a servo string and acts as a device on that string. It receives a device ID as it communicates with the SPU.

The Analog Feedback feature is intended for reading the position of a motor using a potentiometer as a feedback. However the PCU is designed to read any kind of analog input. In this case, the PCU serves as a general purpose analog monitor device.

[1] For more information regarding the Old Servo operation, refer to Servo Micro-controller Documentation Project.

1.2 System Message Flow

Unlike the Old Servos, the PCU is intended to be connected to a dedicated string where the power stays activate all the time.



2. Software

2.1 Overview

The PCU 100 firmware is implemented in C language.

To promote the object oriented methodology in the project, each of the features are implemented in the form of state machines. The PCU 100 firmware uses a Real Time Kernel[2] allowing multitasking operations to be performed.

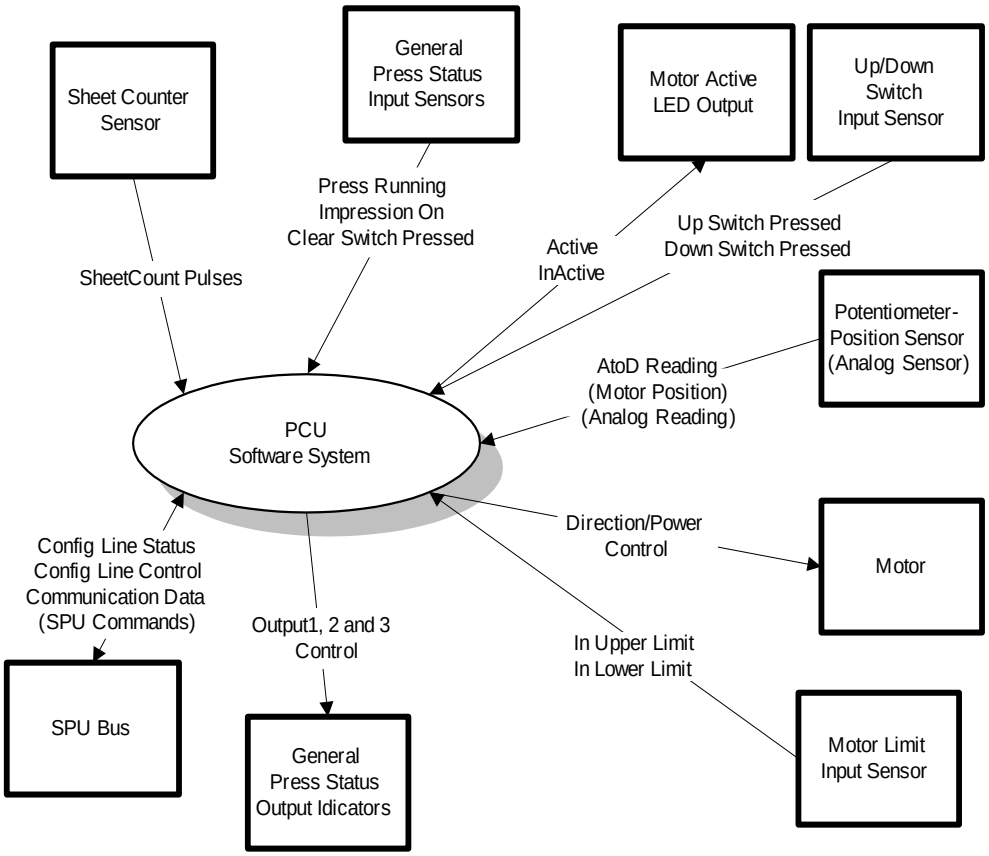
A Kernel Based implementation was chosen :

1. Due to the number of task that have to be performed.
2. Modeling
3. Maintainablitiy
4.

[2] See PCU 100 Real Time Kernel documentation for more information.

2.2 High Level Hardware to Software Interface Diagram

The diagram below shows the relationship of the PCU software to the hardware environment :

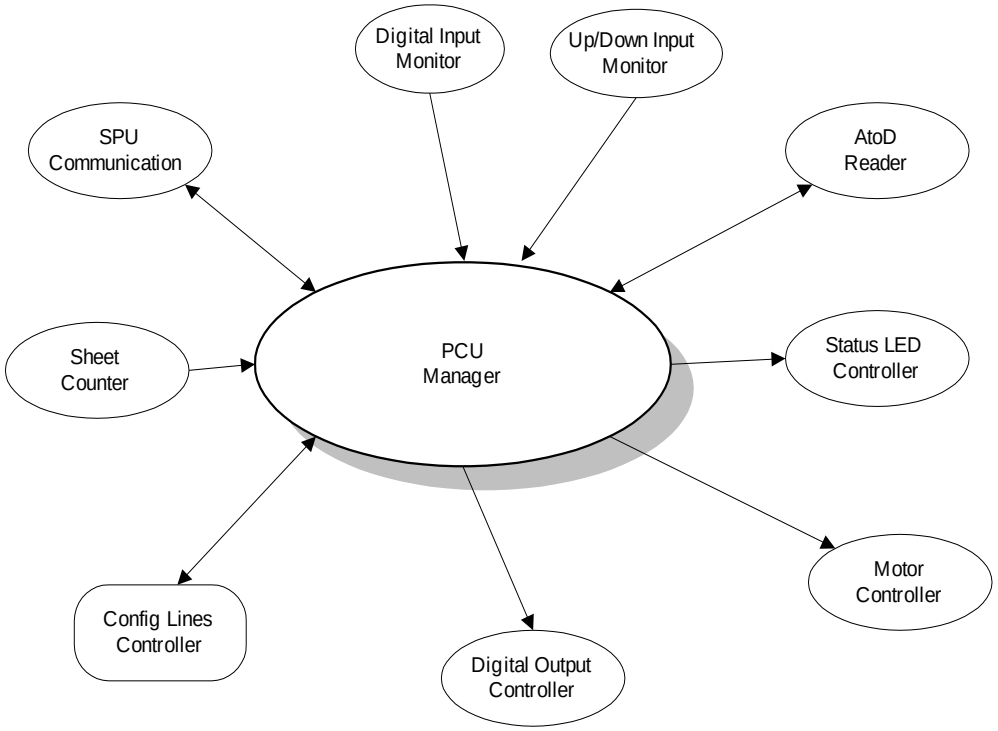


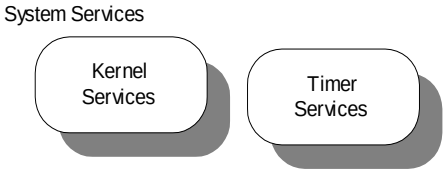
This diagram provides us an overview of how the firmware interfaces to the outside world.

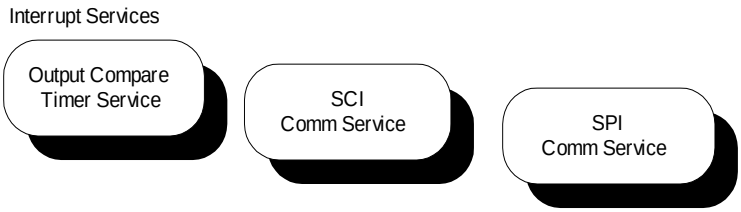
2.3 High Level Software Context Diagram

The diagram below shows the different software components involved in the operation of the PCU 100 system. Each of these components are interfaces to the hardware components as seen in the previous diagram. The relationship between the PCU Manager and it's subordinate tasks can be considered as a Master and Slave relationship where PCU Manager is the master.

Each of these components are further described in detail in the next pages.







Here is a brief description of the jobs that each of the task :

Task Name	Job Description
PCU Manager	Performs the coordination or synchronization of all the tasks within the system.
SPU Communicator	An interrupt driven task that uses the SCI Comm services. This is in charge of all the communication tasks which includes reception and transmission of all messages on the SPU bus.
AtoD Reader	Communicates with the AtoD chip to retrieve its current voltage readings. This machine uses the services of the SPI Comm.
Motor Controller	Performs all the necessary actions to perform all of the motor movements.
Digital Output Controller	Runs at every xx msec. This task updates the state the of the 2 digital output ports.
Digital Input Monitor	Runs at every xx msec. This task scans the 5 digital inputs (including the Up/Down Input) to determine their states.
Config Line Controller	Controls the states of the Config Line.
Status LED Controller	Running at every XX msec, this updates the state the Status LED. The Status LED provides several different cadences to indicate several different states of the PCU operation. See …
Sheet Counter	With a given scan rate, this machines monitors the Sheet Count input and counts the number of pulses it detects.
Kernel Services	Provides a basic timing service for each of the machines as well as the ability for each machines to communicate with each other.
Timer Services	Provides a basic timing service of 1 msec or the “System Tick”. The state machines uses this timer to perform their tasks at given intervals.

2.4 PCU Manager Machine (PMM)

Description :

As seen in the High Level Context diagram, PMM is in charge of the overall system synchronization. To ensure the authority of PMM, communications between the subordinate state machines are strictly enforced to go through PMM as a rule.

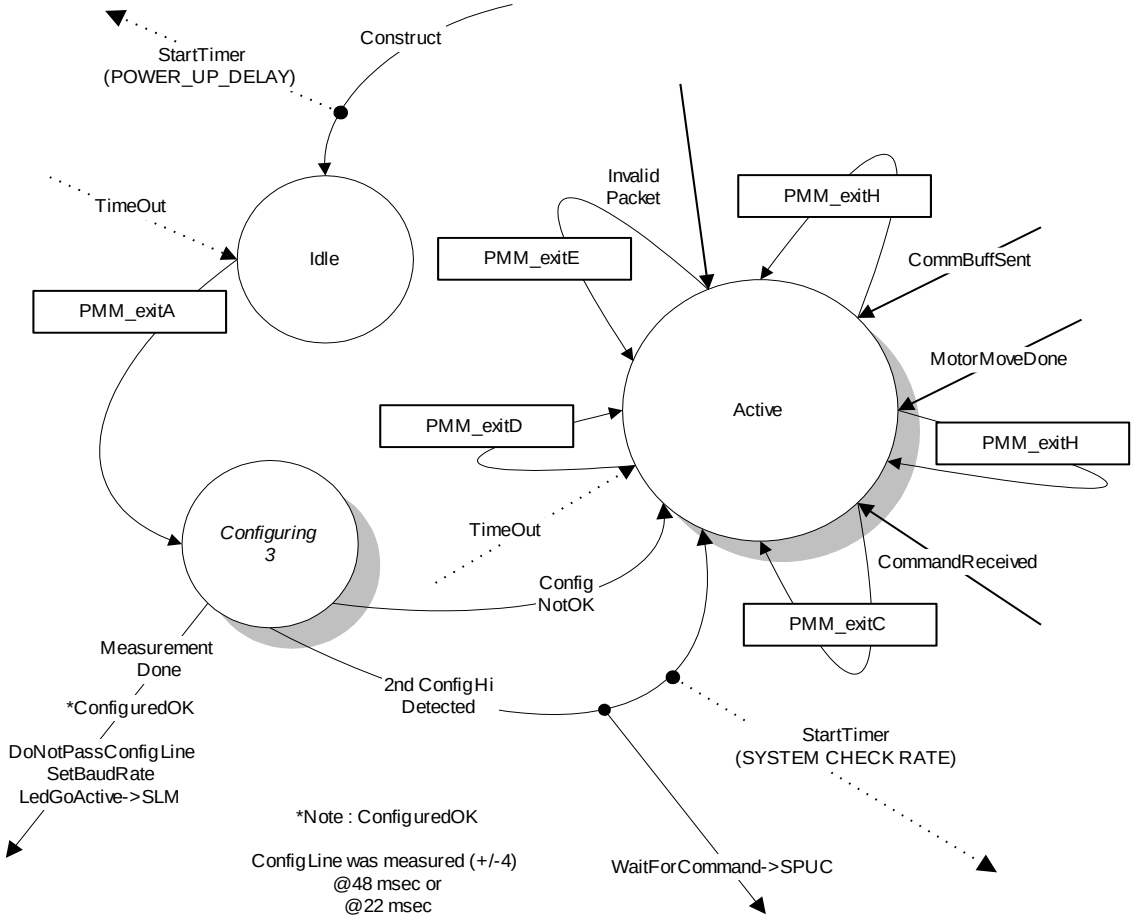
2.4.1 PMM State Transitions, General

PMM goes through 3 basic states. These are Idle, Configuring and Active state. The Idle state primarily waits for the POWER_UP_DELAY timer to expire allowing the system to stabilize. Configuring is the state where the PCU determines the BAUD Rate Setting by monitoring the Config Line.

When configuration passes, PMM goes into the Active state where it runs these 3 general tasks concurrently :

1. Responds to execute SPU commands
2. Performs system background task
3. Responds Local Positioning Events (Up/Down Switch)

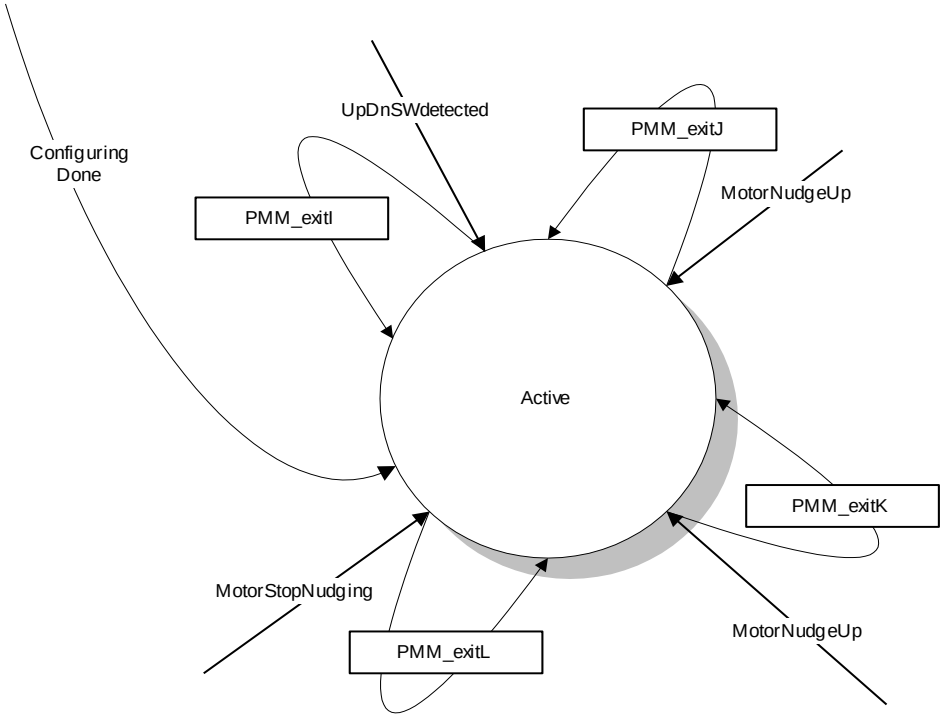
The diagram below shows the transitions that occur when either one of the first 2 general tasks is executed:



2.4.2 PMM State Transitions, Local Positioning Events

Since PMM is in charge of the overall system operation it is expected to receive several messages from its subordinate state machines. In addition to the messages shown in the previous diagram, Local Positioning message is another group of messages that PMM can process.

Local Positioning is shown here separately to indicate the messages that are processed when Local Positioning is initiated. The diagram below shows the transitions that occur when PMM responds to the Up/Down Switch events.

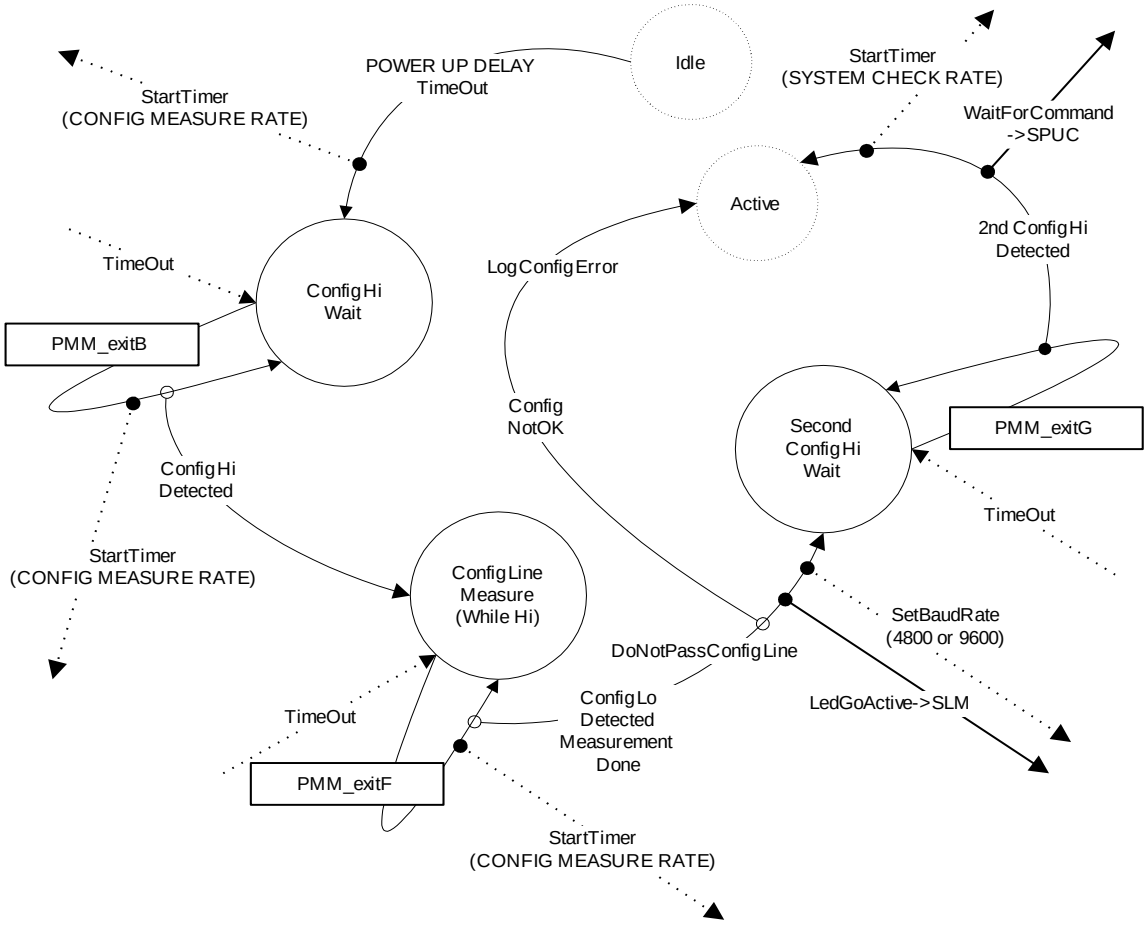


2.4.2.1 PMM State Transitions, Configuring State

Upon power up, the very first signal expected from the SPU is the Configuration Pulse. After some power up delay, PMM is initiated to be in the Configuring State to handle this Configuration Pulse event.

In general the Configuring State is simply the state where the Config Pulse is monitored, captured and measured. As stated earlier the Config Line provides a reference for the PCU to determine the BAUD rate at which to communicate. The PCU will not be able to communicate until a successful configuration process happens. Consequently, all system functions are disabled before the PCU can communicate.

The following diagrams show the states in detail while in PMM is in the Configuring State.



Note :

Configured OK means that a Config pulse of 22 msec or 48 msec was detected. An error window of +/- 4 msec is provided. For more information regarding the Configuration Timing protocol see Baud Rate Configuration on page 49.

2.5 SPU Communication Machine (SPUC)

After a successful configuration is achieved, the very first subordinate state machine that is made active is the SPU Communication Machine.

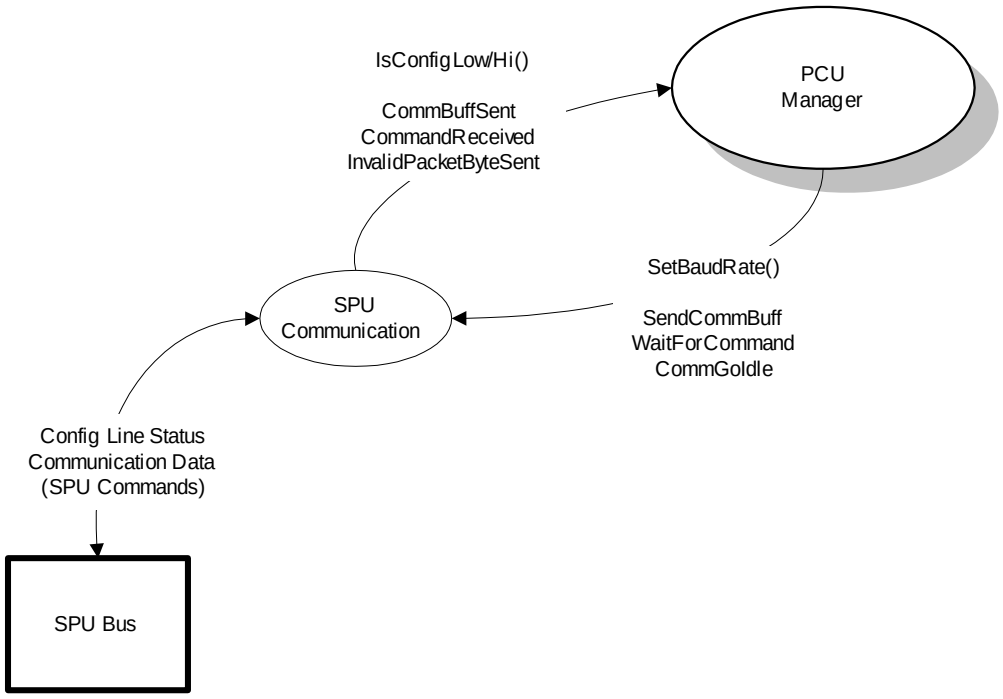
SPUC performs all of the communication work for the system. This state machine is aware of the data format protocol that the PCU is expected to receive and process. It also knows about the timing protocol and independently performs all of the necessary communication timing requirements.

2.5.1 SPUC High Level Context Diagram

Overview :

SPUC receives and sends data through the services of the SPU Bus hardware. SPUC uses the SCI (Hardware UART) feature of the processor to perform its tasks.

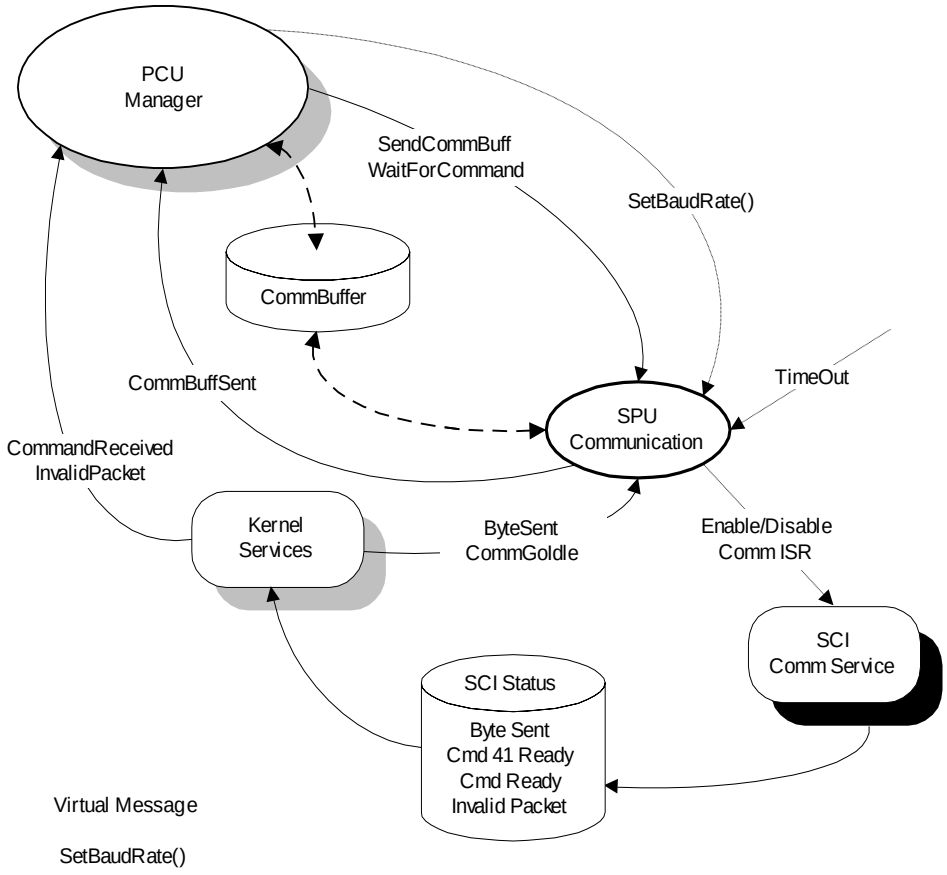
The Config Line has another special purpose for the system. Aside from providing a reference pulse for the Baud Rate Configuration it also signals the start of a new data packet coming from the SPU. SPUC uses this signal to synchronize itself to the incoming data.



2.5.2 SPUC System Interaction

Due to the interrupt nature of SPUC, the Kernel has been designed to interface between the COMM ISR and the high level state machines (PCUM and SPUC). This diagram is somewhat different than the previous one since it shows more of the implementation aspect of the SPUC and PCUM interaction.

Within SCI Comm Service (ISR), flags are setup for the PCU Kernel to indicate different conditions and events that occur. The Kernel then processes and translates these flags into messages which are then relayed back to their respective receivers.



Note :

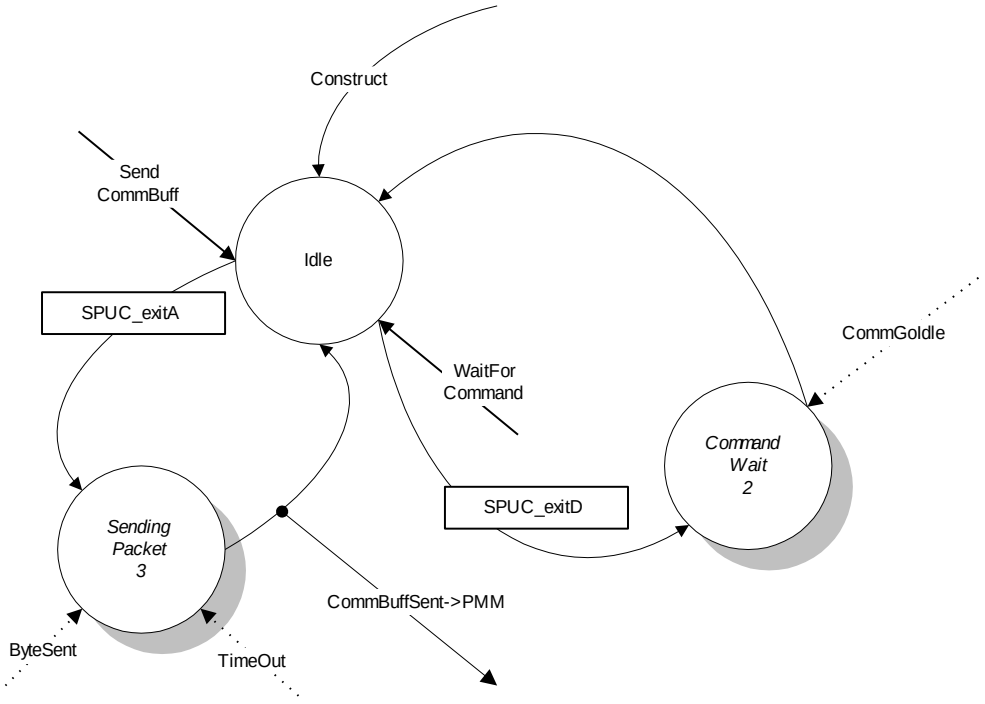
PCUM and SPUC shares the Communication Buffer. Since PCUM is the master, access conflict to the Communication Buffer resource is avoided.

2.5.3 SPUC State Transitions

SPUC only have 2 general active states :

1. Sending a Packet
2. Waiting for Command

These states are further shown in detail in the next pages.

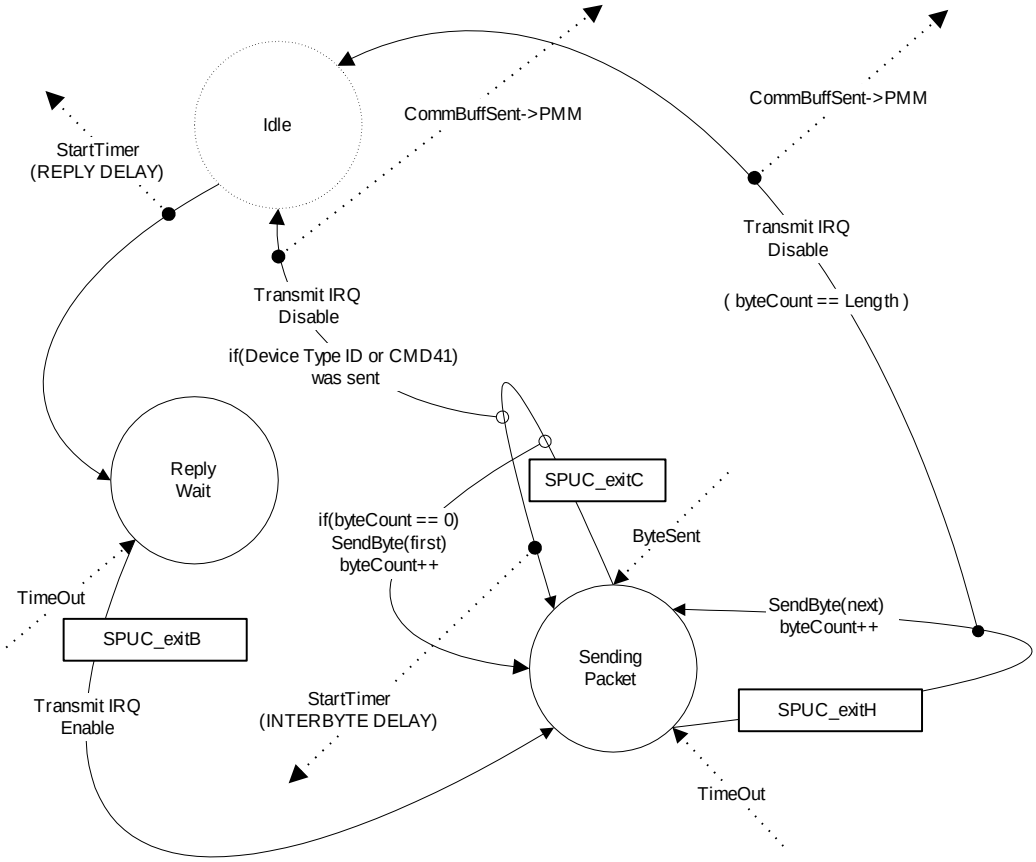


2.5.3.1 SPUC State Transition, Sending Packet

While sending states SPU goes through 2 states.

Reply Delay is when the state machine waits for the Reply Delay Timer to expire. This Reply Delay timer is a constant based on the communication timing protocol. Once the timer expires SPUC begins the transmission.

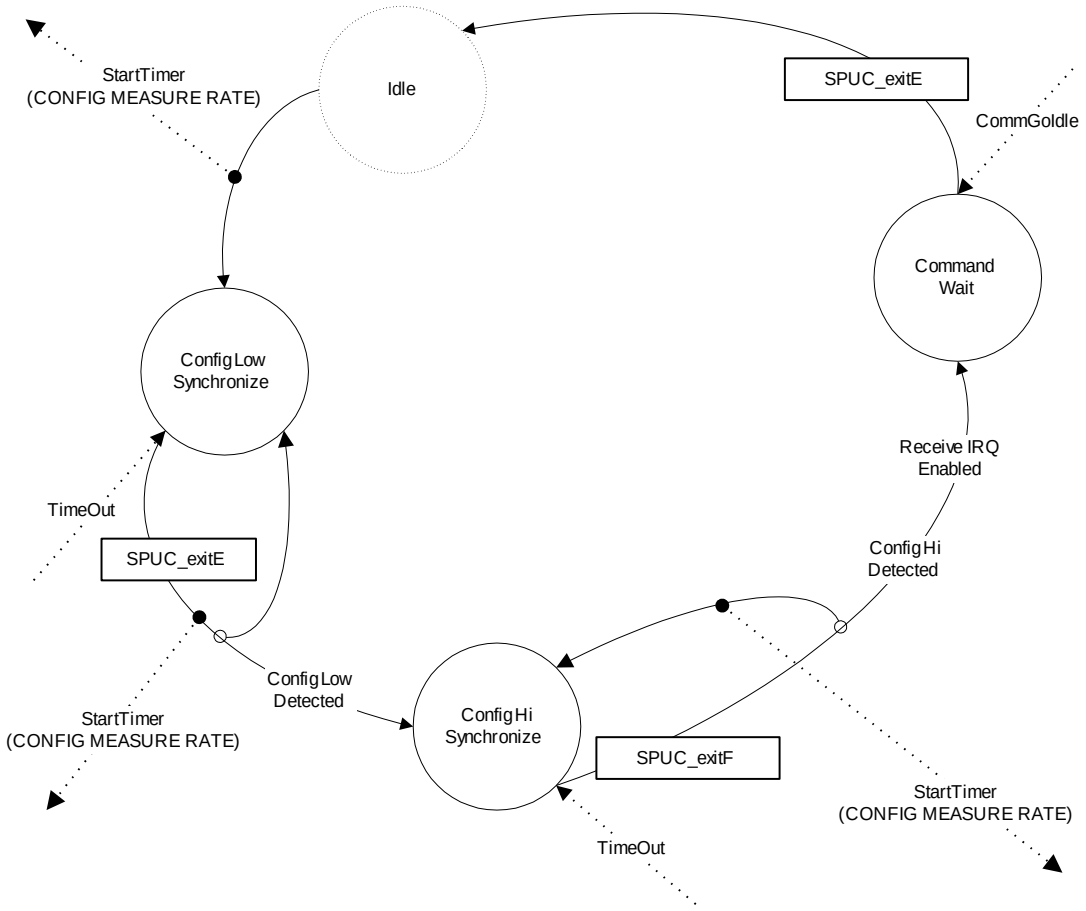
Each byte is sent until the whole packet is completely sent. In between the byte transmissions, a 2 msec delay is injected (Inter-Byte Delay). This is again to conform to the communication timing protocols.



2.5.3.2 SPUC State Transition, Command Wait

When SPUC receives WaitForCommand message, it immediately goes into the Config Detection States. This allows SPUC to synchronize to the start of the new message. If the SPU is in the middle of a message transmission when SPUC.

Once SPUC is synchronized to the beginning of the message packet, the SCI Comm Service (ISR) performs all the work. SCI Comm analyzes the validity of the incoming data or packet received and flags the Kernel regarding the result of the analysis. The Kernel then relays this result to PMM and instructs SPUC to go back to the IDLE state. PMM reacts accordingly.



IMPORTANT :

At any point while SPUC is in Command Wait, a GoIdle message can be received forcing the state machine to stop running. SPUC is consequently re-synchronized when the next WaitForCommand message is received.

The GoIdle command is sent to SPUC by PMM every time an SPU command is executed. The PCU will not be communicate during the time that the command is being executed. Commands such as the “Move To Position” requires very accurate timing services. Putting SPUC in IDLE was included in the design to minimize any latencies caused by the Communication Interrupts.

2.6 Motor Controller Machine (MCM)

Description :

The Motor Controller Machine is in charge of all the operations required to position the motor.

The “nudge” is another service provided by MCM. This provides a way of making finer motor moves which is necessary in certain situations. The term “nudge” is synonymous to the term “putt” used to describe the tiny moves made by the old servos.

These are the 4 motor positioning services provided by MCM :

1. **Move To Target** - In general, positioning the motor using this service includes deciding the direction at which the motor should turn, applying the power and determining if the target has been reached. This service is normally used to achieve a given position as quickly as possible, while taking advantage of the characteristics of the motor moving at full speed.

2. **Nudge To Target** - This is very similar to the first service in many respects. The only difference is that instead of applying full power to the motor, MCM applies small pulses of power with a known duration until the target is reached.

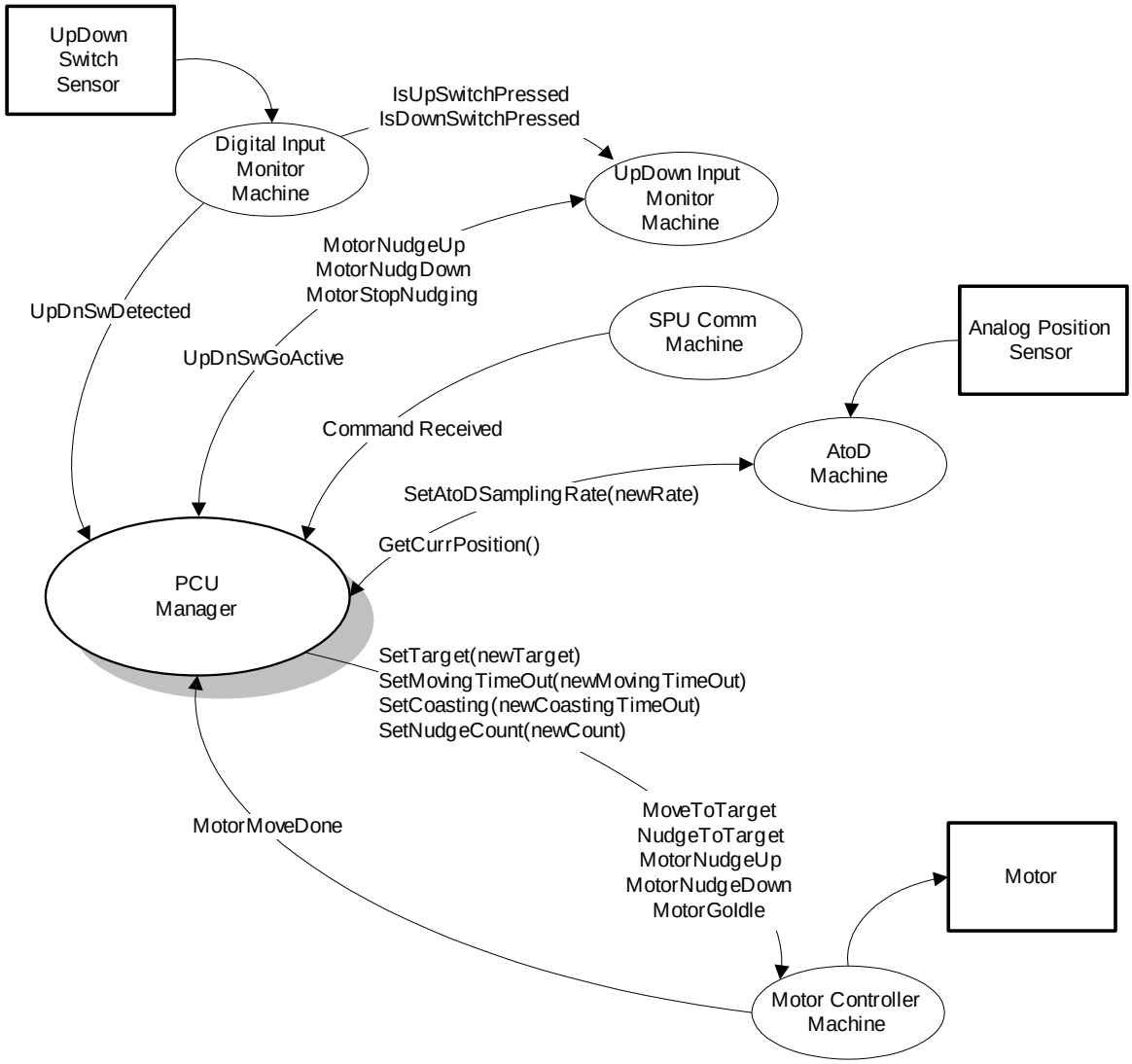
3. **Nudge Up/Down** - This service also does not require a target to be set. This service will cause MCM to apply 1 pulse of power (with a known duration) in a given direction.

4. **Nudge X Times** - Like the Nudge Up/Down service, this also does not require a target to be set. Instead, MCM is instructed to apply a known number of pulse (with a known duration). This effect can be achieved by performing these 2 sequences of action (1) Setting the Nudge Count and (2) Sending a Nudge Up or Nudge Down message.

Note : The duration of the nudge pulse is adjustable. See .

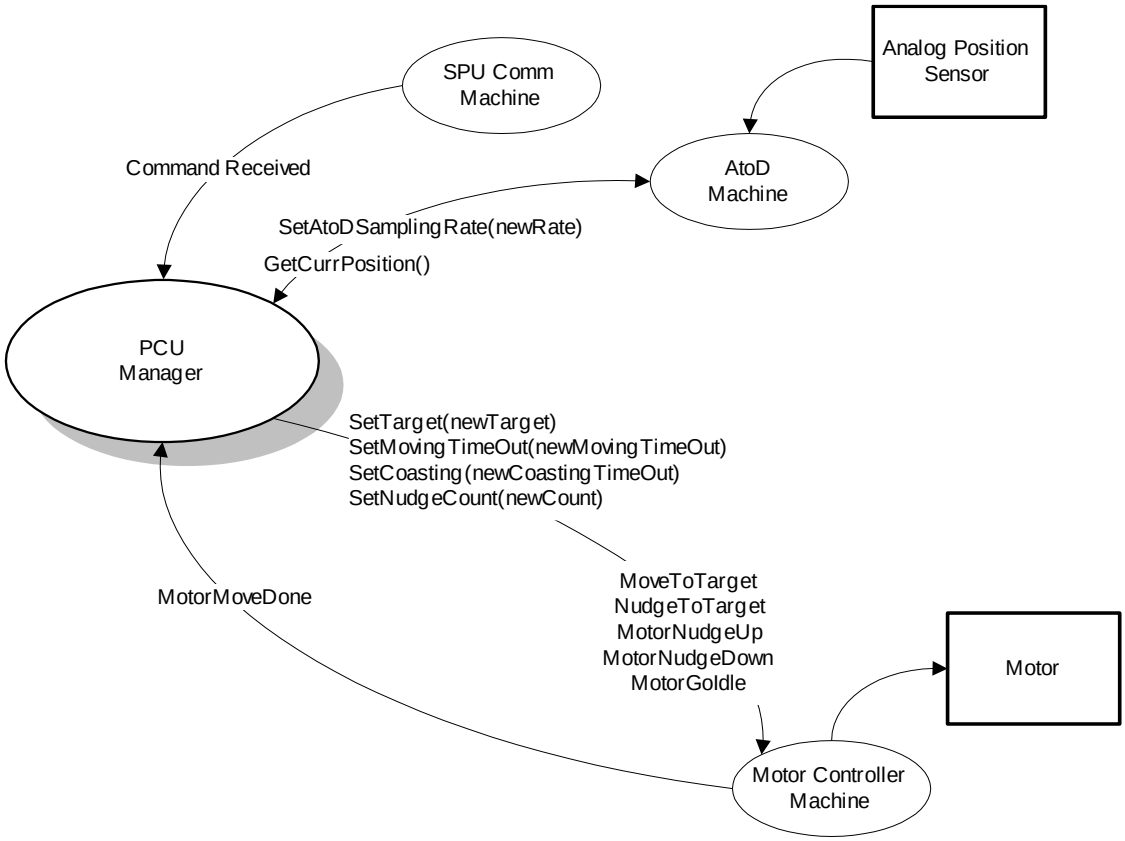
2.6.1 MCM High Level Context Diagram

As seen below, MCM is involved with several other machines are to perform this task of positioning the motor. The diagram below shows the state machines as well as the messages that are used in the operation of the Motor Controller Machine.



2.6.2 MCM System Interaction, Move To Target

These are the specific state machines and messages that are involved when the Move To Target and Nudge To Target command is executed.



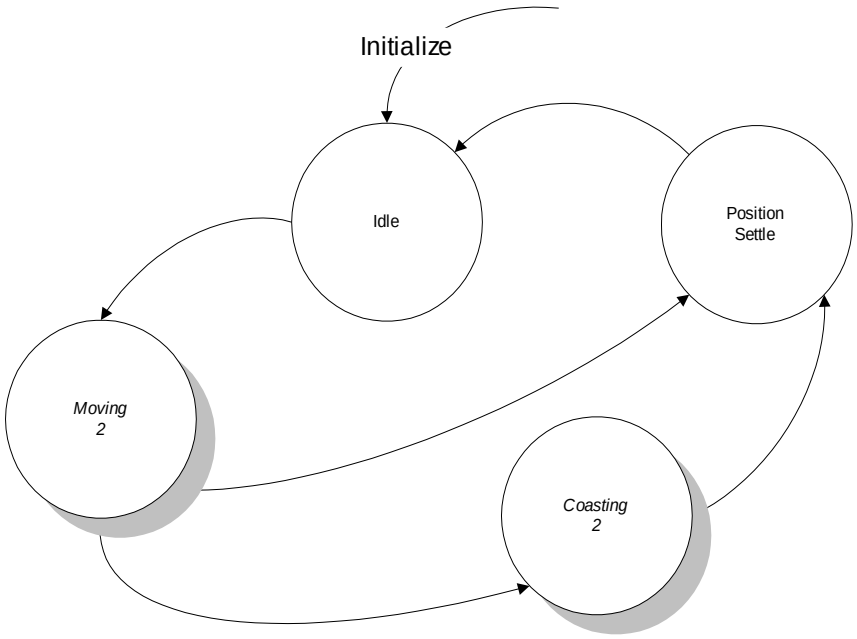
Note :

The AtoD Machine provides the position feedback for the Motor Controller Machine when determining the motor position.

2.6.3 Moving to Target States

Description :

When the PCU Manager receives a “Move to Target” command from the SPU Comm machine, the Motor Controller Machine is instructed to perform the moves required to achieve the desired position. These are the 4 states that the servo the motor goes through when it is executing the Move To Target command:

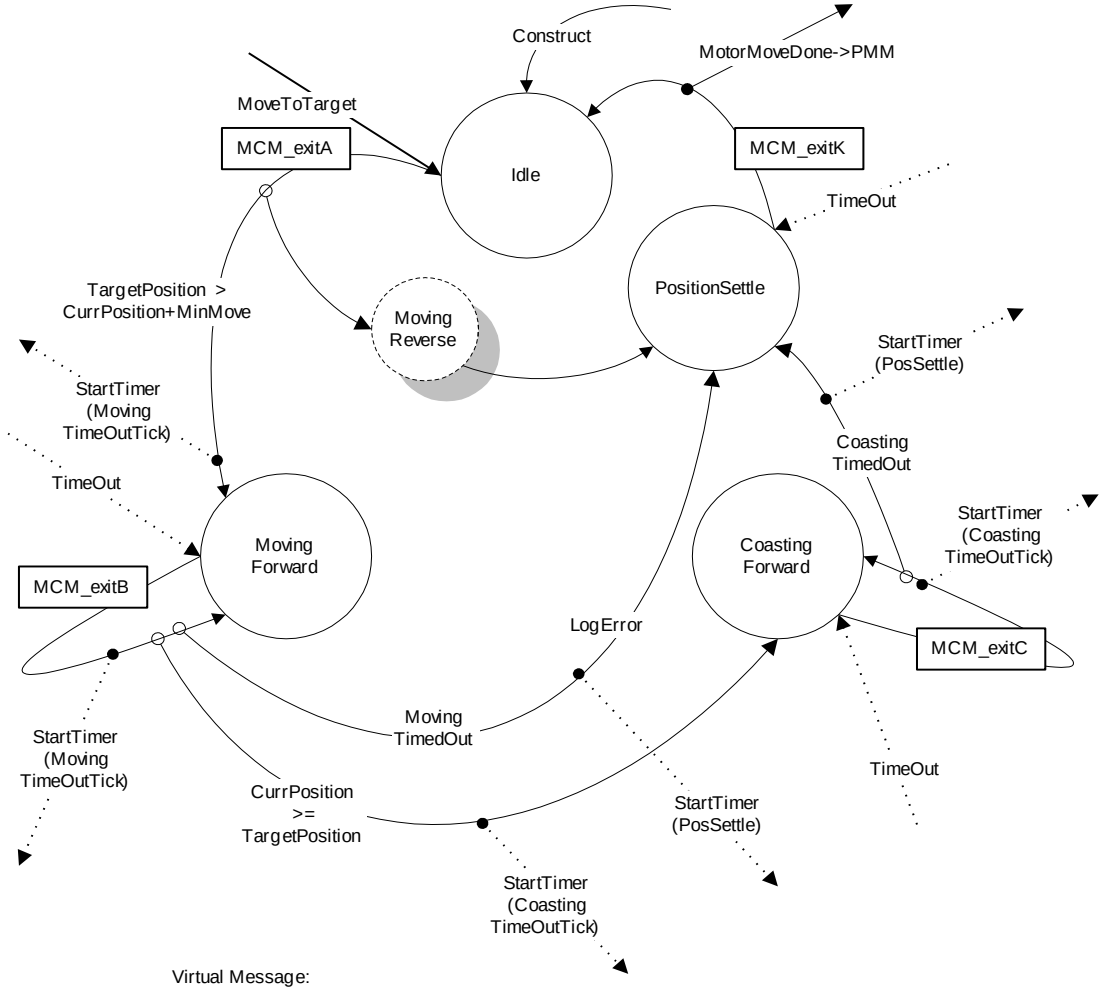


Upon power up, MCM is placed in the idle state.

When a move command is received, the motor is initiated to move in a given direction This is considered the Moving State. When the servo reaches its target position, it then goes into the Coasting State. After the motor is done coasting, the machine goes into a settling state. This state was created to allow the AtoD readings to stabilize before the machine sends a message, telling the PCU Manager (PMM) that it has completed the move command. These states are shown in detail in the next pages.

2.6.3.1 Forward Moving and Coasting State Transition

These are the different states when the motor is moving in the reverse direction to be positioned :



Virtual Message:

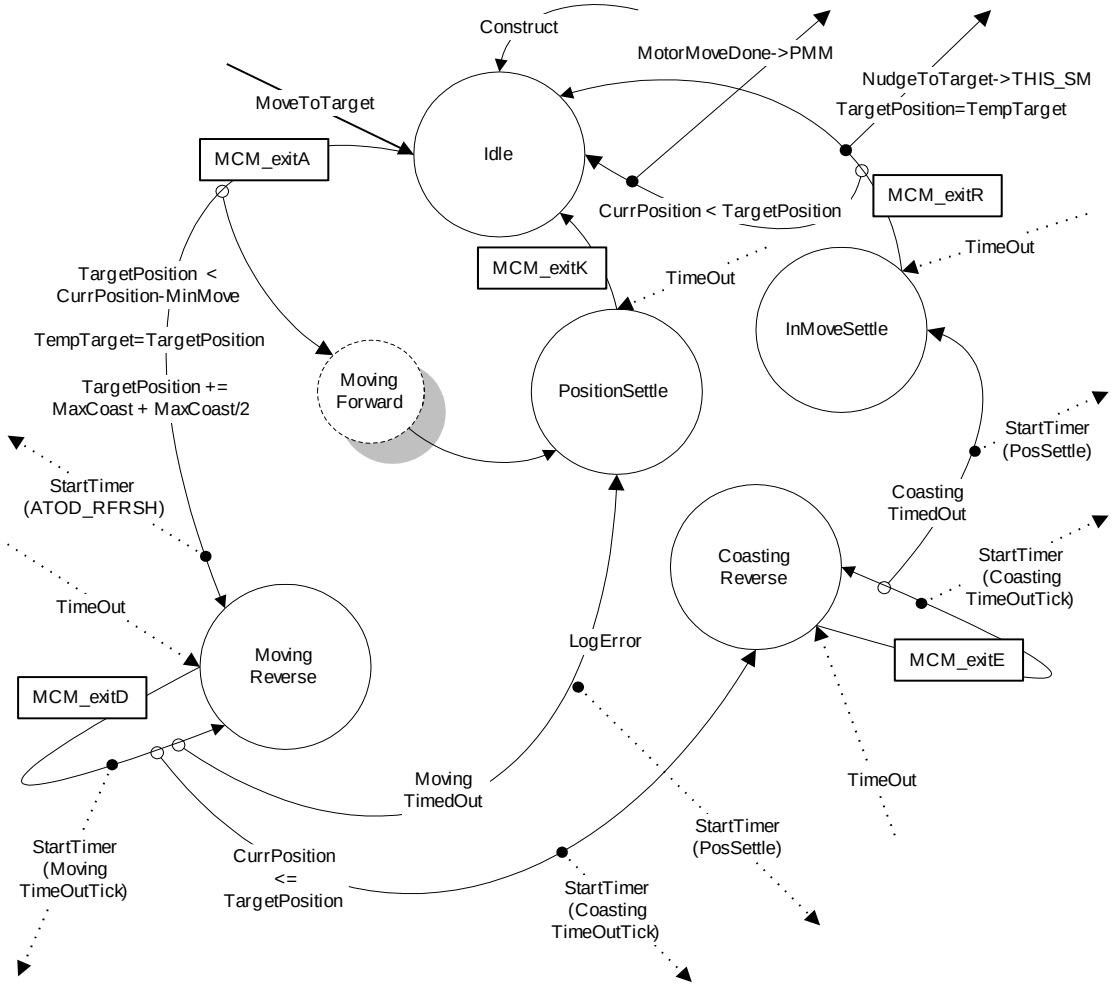
SetTarget(newTarget)
SetMovingTimeOut(newMovingTimeOut)
SetCoasting(newCoastingTimeOut)

Moving TimeOut : Is determined when the motor has stopped moving for a given amount of time and the target was not yet reached. The error is logged in this case and reported back to the SPU.

Coasting TimeOut : Is determined when the motor has stopped moving for a given amount of time after the power has been cut off. The difference between the target and the actual final position is considered the coast distance.

2.6.3.2 Reverse Moving and Coasting State Transition

These are the different states when the motor is moving in the reverse direction to be positioned :



Note :

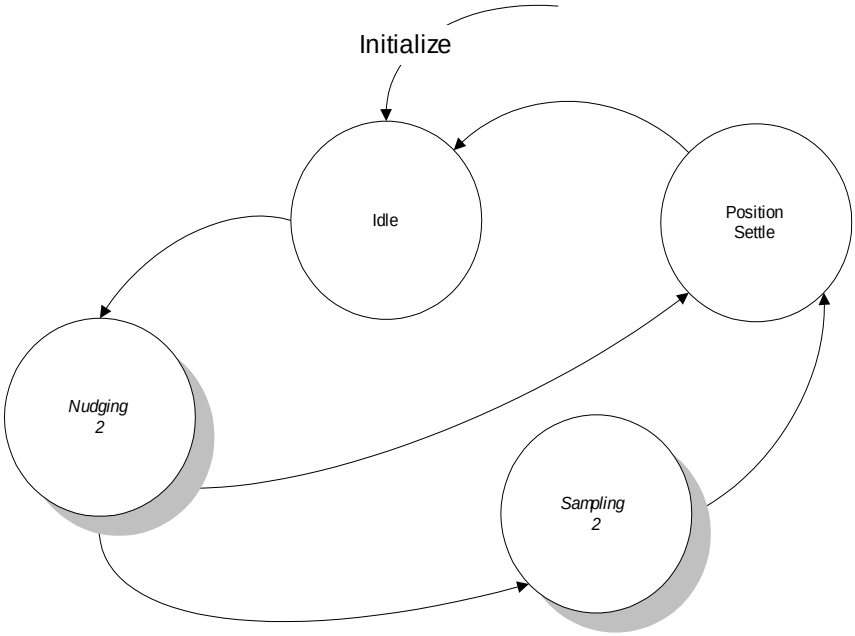
The states are identical for both instances where the motor is moving forward or moving reverse to be positioned. The only difference is that the exit function that are executed. These exit functions differ only in way the target is determined to be reached. For example :

MCM_exitD uses the expression : CurrPosition <= TargetPosition where as
MCM_exitB uses the expression : CurrPosition >= TargetPosition.

2.6.4 Nudging to Target States

Description :

When the PCU Manager receives a “Nudge to Target” command from the SPU Comm machine, the Motor Controller Machine is instructed to perform the moves to achieve the desired position. These are the 4 states that the servo the motor goes through when it is executing the Nudge To Target command:

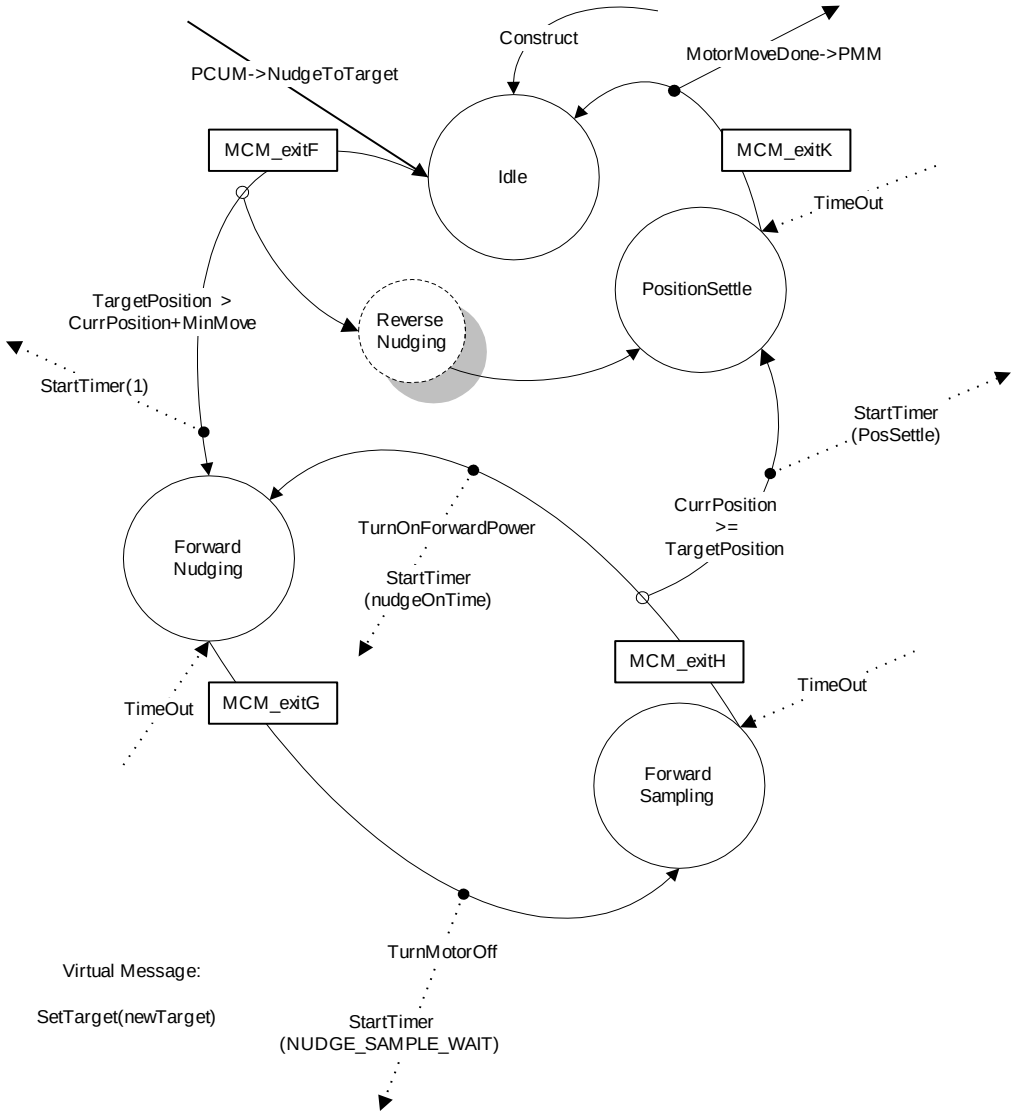


When a nudge to target command is received, the motor is initiated to move in a given direction. This is considered the Nudging State. This is the state where 1 pulse of power is applied. After the nudge pulse timer expires, the power is cut off and MCM goes into the Sampling State. In this state, MCM takes several samples of the AtoD reading to get the motor’s current position. When the servo reaches its target position, it then goes into the Position Settle State and then back to Idle. A message is sent telling PMM that the motor positioning is done. These states are shown in detail in the next pages.

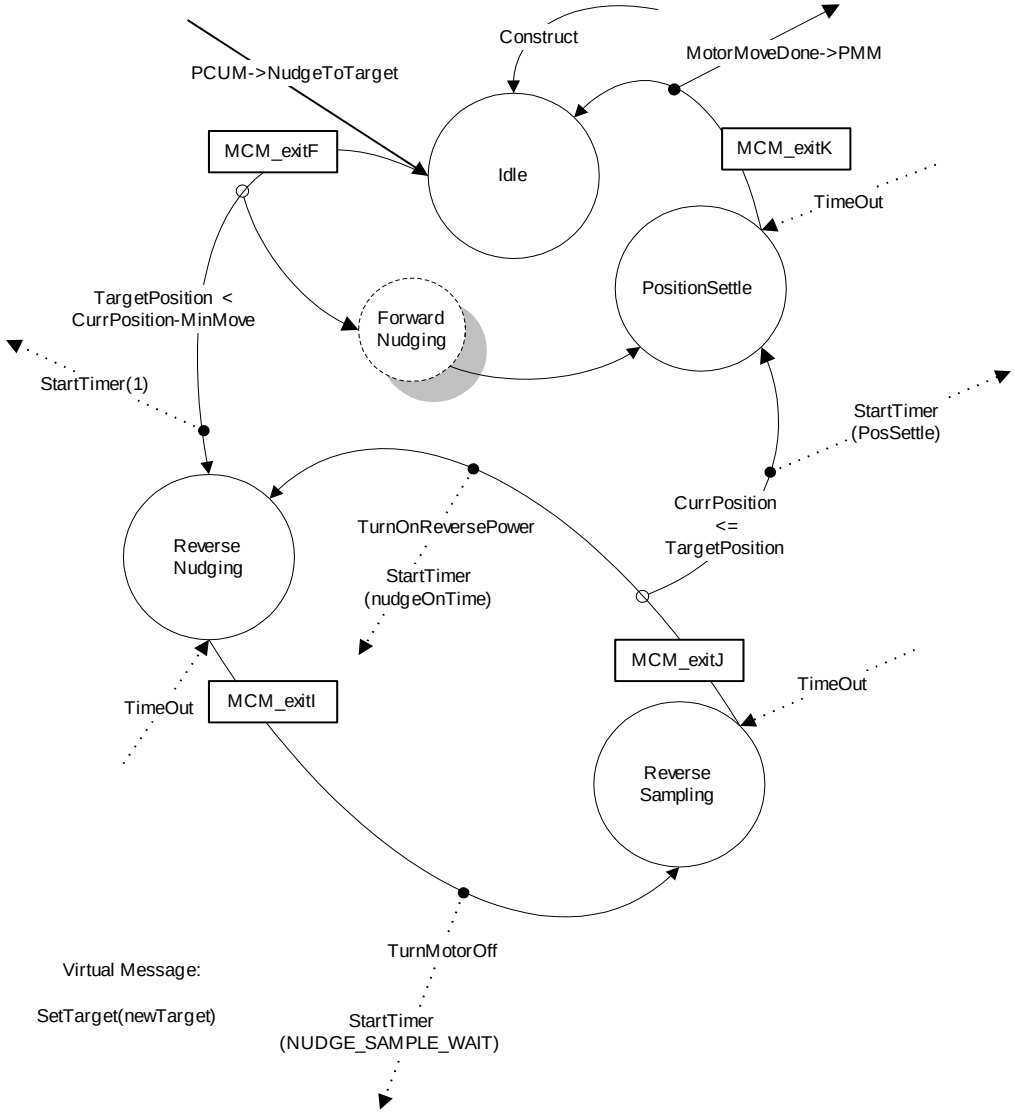
Note :

All states pass through the Position Settle state prior to going back to IDLE. This is to ensure that PMM is informed that MCM has completed its job.

2.6.4.1 Forward Nudging To Target and Sampling State Transition

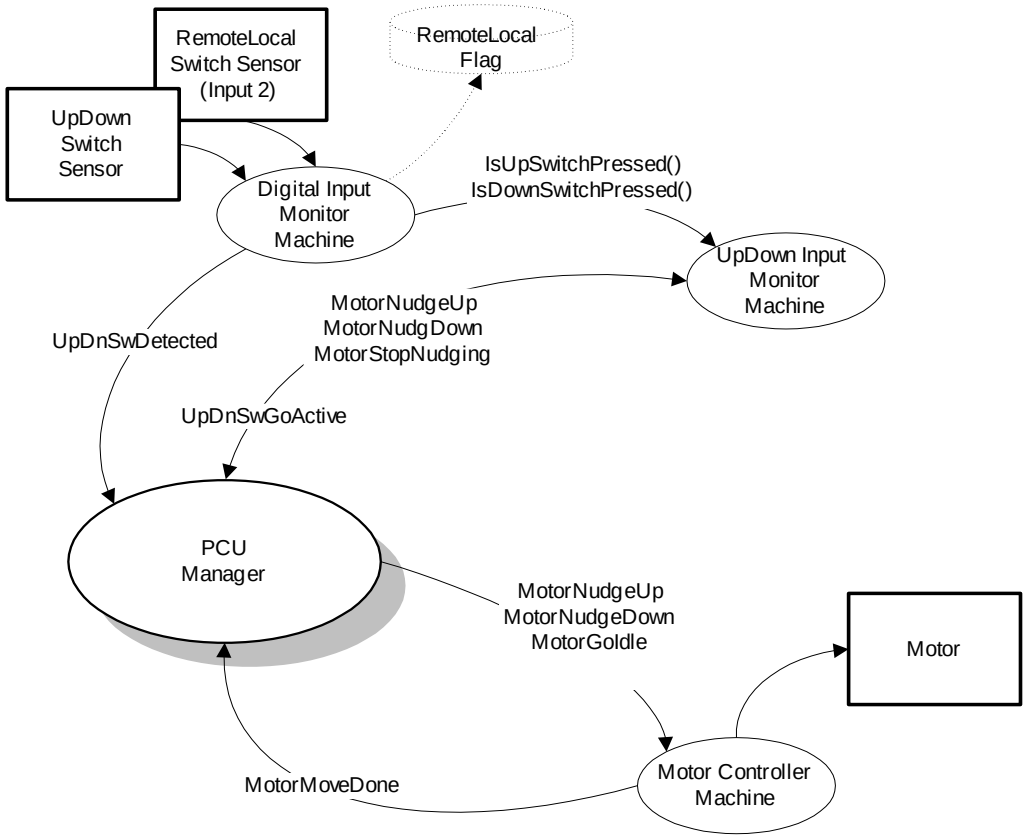


2.6.4.2 Reverse Nudging To Target and Sampling State Transition



2.6.5 MCM System Interaction, Nudge Up/Down

Nudge Up/Down refers to the process executed when local positioning is initiated. These are the main components involved when the Nudge Up/Down is being executed :



The Nudge Up/Down events are generated by the activation of the local position control switches.

The Up/Down switch inputs are part of the DIM (Digital Input Monitor state machine). When either one of the Up/Down switch is detected to be active, a message is sent to PMM (PCU Manager) to inform it about the event. PMM then decides how to pass this event to MCM (Motor Controller) to generate the Nudge.

The Up/Down Input Monitor Machine plays a special role in handling the Up/Down switch events after the initial detection. This machine is described later in detail. See “Up Down Input Monitor Machine (UDM)” on page 38.

The Local/Remote Switch input provides a means of enabling or disabling the Local Positioning Control. This input is normally set in the Remote Mode thereby making the Local Controls normally inactive. The Local Remote Flag which is a global variable is updated every time Local/Remote Switch input changes state.

IMPORTANT :

A flag is set to inform the SPU/Console regarding the activation of the Local Controls. The SPU can then decide how to handle this scenario.

The SPU/Console also has the ability to monitor the Local/Remote Switch Input status to decide :

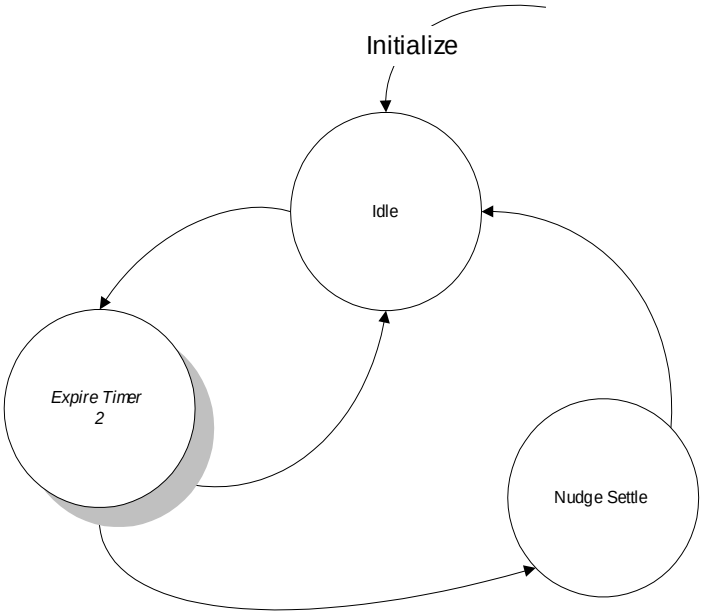
a) When to Update Console Values based on current PCU Position readings
b) When it is allowed to send a Move command (Remote Mode)
c) and when it is not (Local Mode)

2.6.6 Nudging Up/Down States

Description :

When PMM receives the UpDnSwDetected message, it forces MCM to Go to IDLE if it is not in IDLE. This means that the very first Up/Down Switch depression only causes a moving motor to stop and does not create the Nudge. Only when the Motor is IDLE can the Nudge be generated.

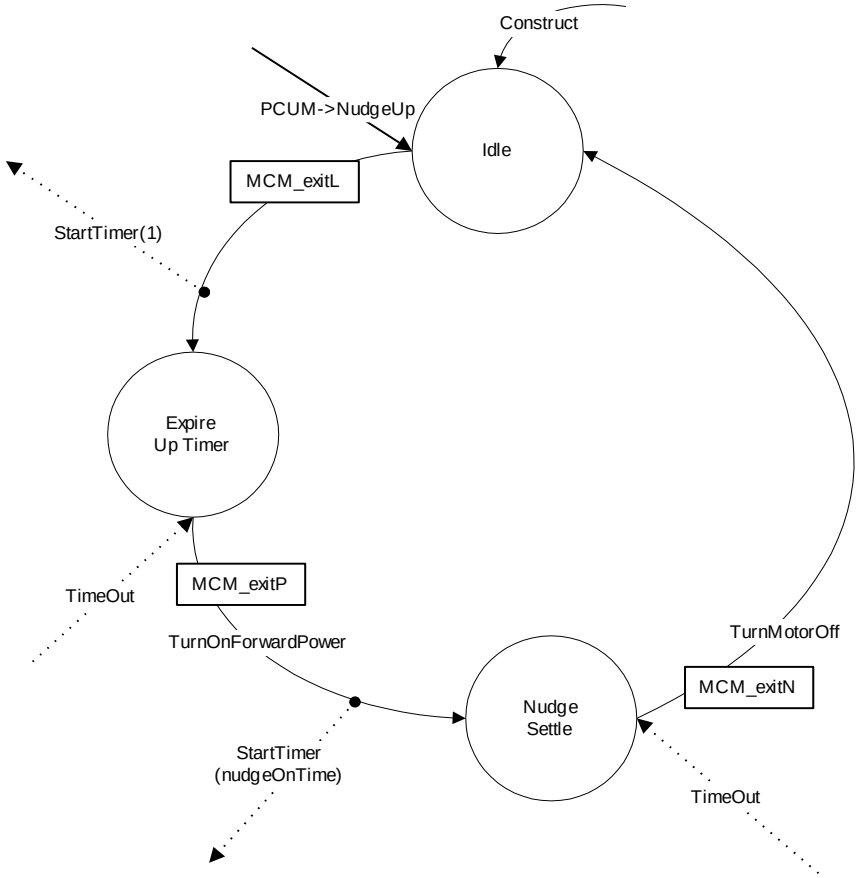
These are the different states that MCM receives the Nudge message from the IDLE state.



When a Nudge Up or Down Command is received, MCM first goes into the Expire Timer state. Since the Nudge is a time-based motor move, this state was created to ensure that timing latencies are minimize. This also causes the nudge pulse to be more consistent. There are 2 instances of the Expire Timer state. One for Nudge Up and one for Nudge Down.

2.6.6.1 Nudging Up State Transitions

This diagram shows the different messages used and the different states that MCM goes through when executing the Nudge Up command.

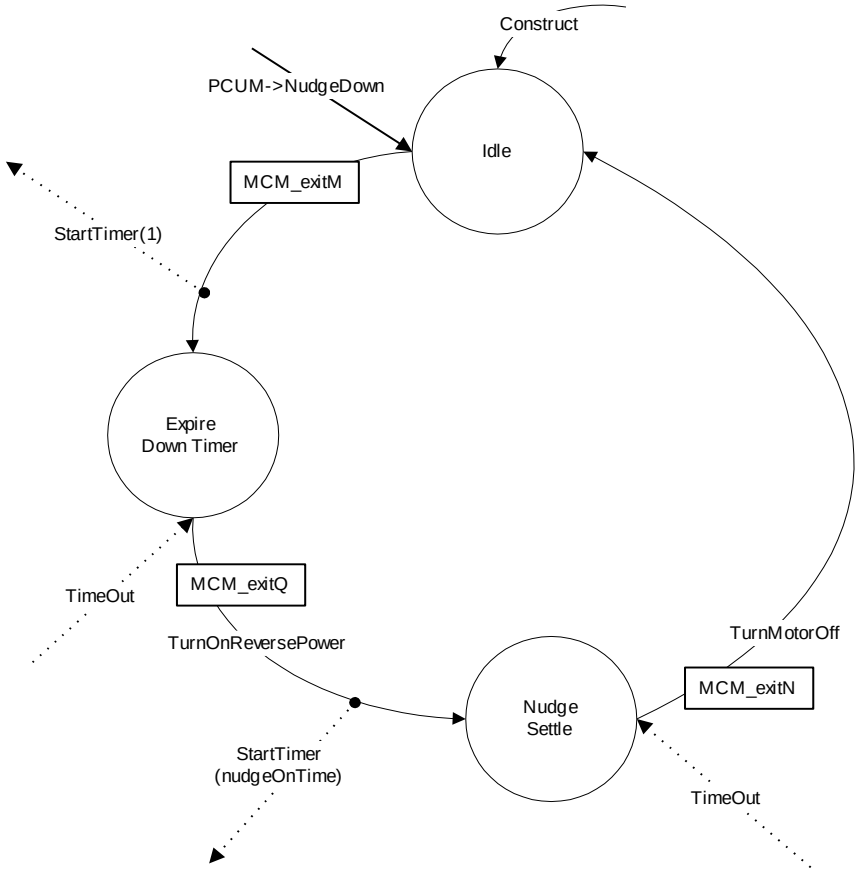


Note :

In between Expire Up Timer state and the Nudge Settle state is where the Nudge pulse is generated. The machine goes through the Position Settle state to make sure the AtoD readings are stable by the time PMM is informed about the completion of the Nudge. This is to make sure that SPU gets an accurate position should it request for the current position after the Nudge event.

2.6.6.2 Nudging Down State Transitions

This diagram shows the different messages used and the different states that MCM goes through when executing the Nudge Down command.

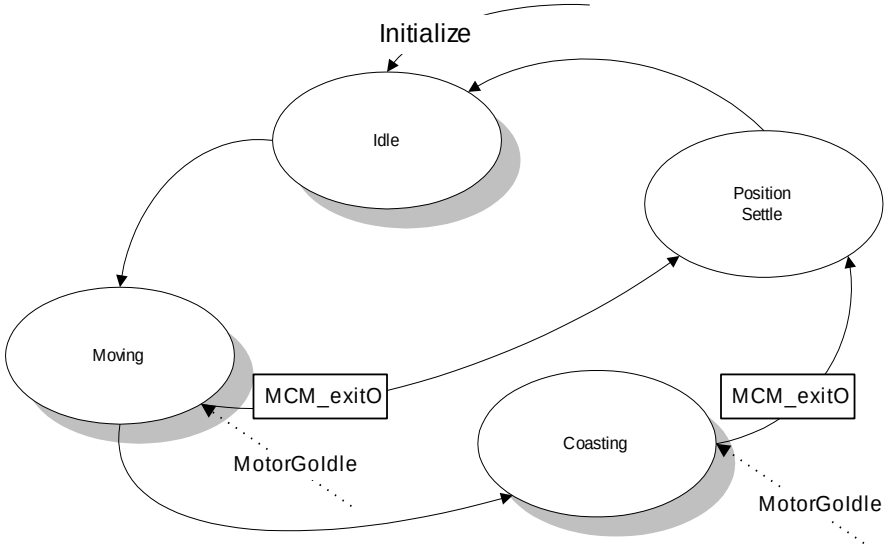


2.6.7 Motor Go Idle

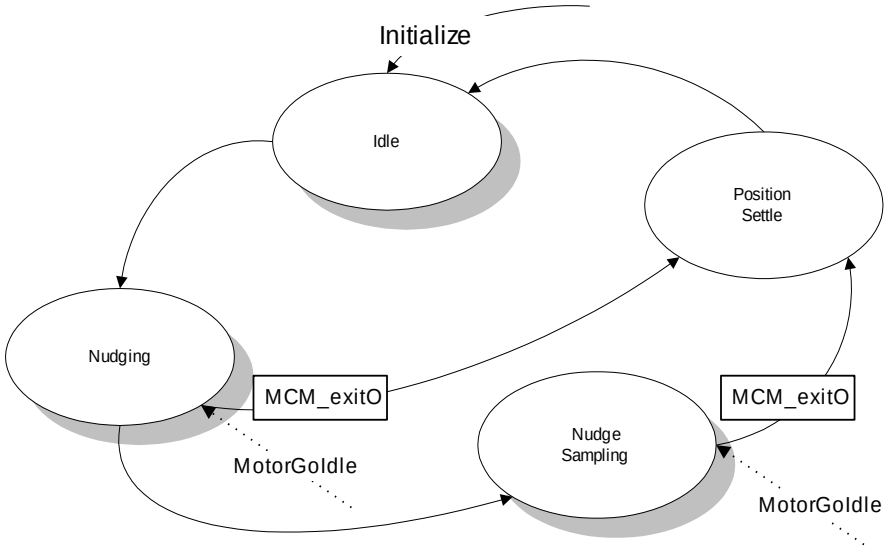
Description :

There are some cases where the PCU Manager can decide to cancel a motor move that is currently running. A “Go Idle” message has been provided to allow the Manager to do just this. At any point while the motor is in moving whether it is coasting or in a nudging state, a message can be sent to instruct the machine to go idle. The diagram below shows this operation.

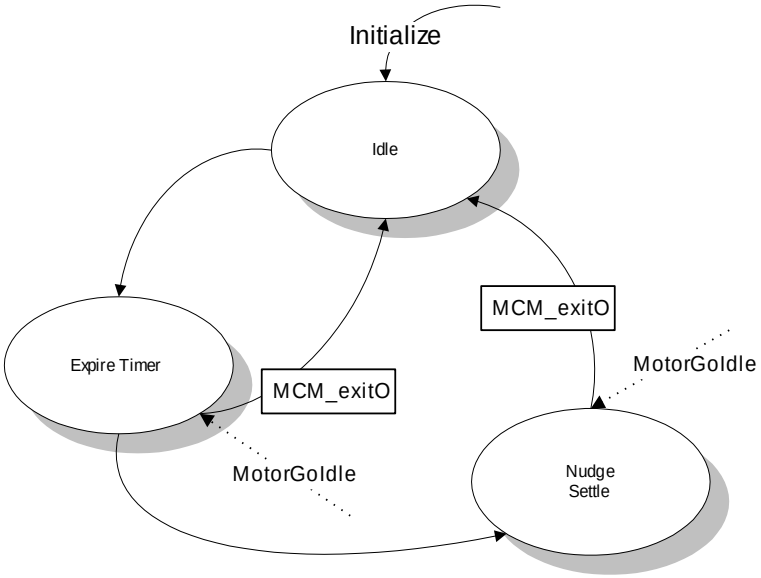
Moving and Coasting States :



Nudging To Position States :



Nudging States (Local Nudging or Nudge X Times) :



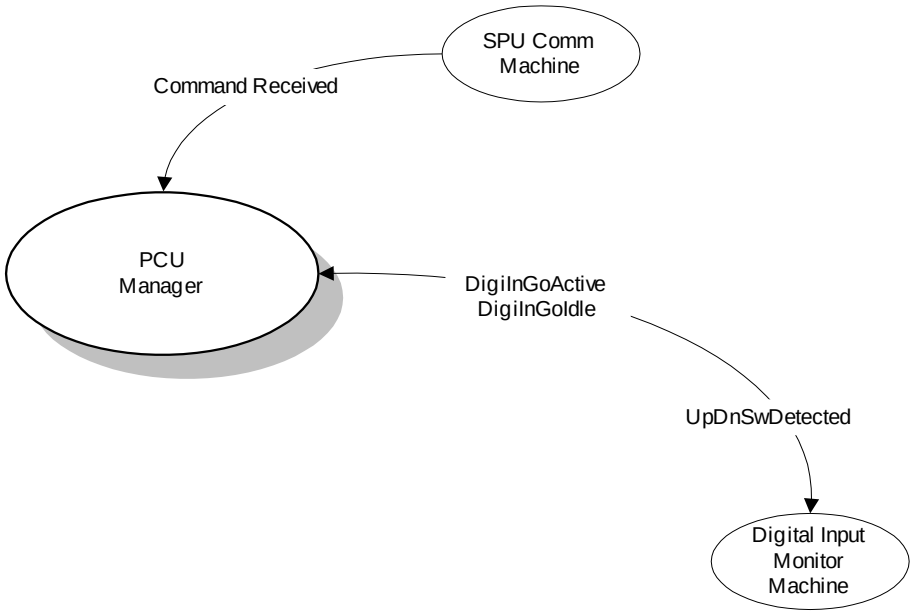
2.7 Digital Input Monitor Machine (DIM)

Description :

The PCU provides 5 Digital Inputs. The Digital Input Machine is in charge of monitoring 3 general inputs as well as the Up/Down switch.

2.7.1 DIM System Interaction

Below is a diagram showing how DMM interacts with the rest of the system :

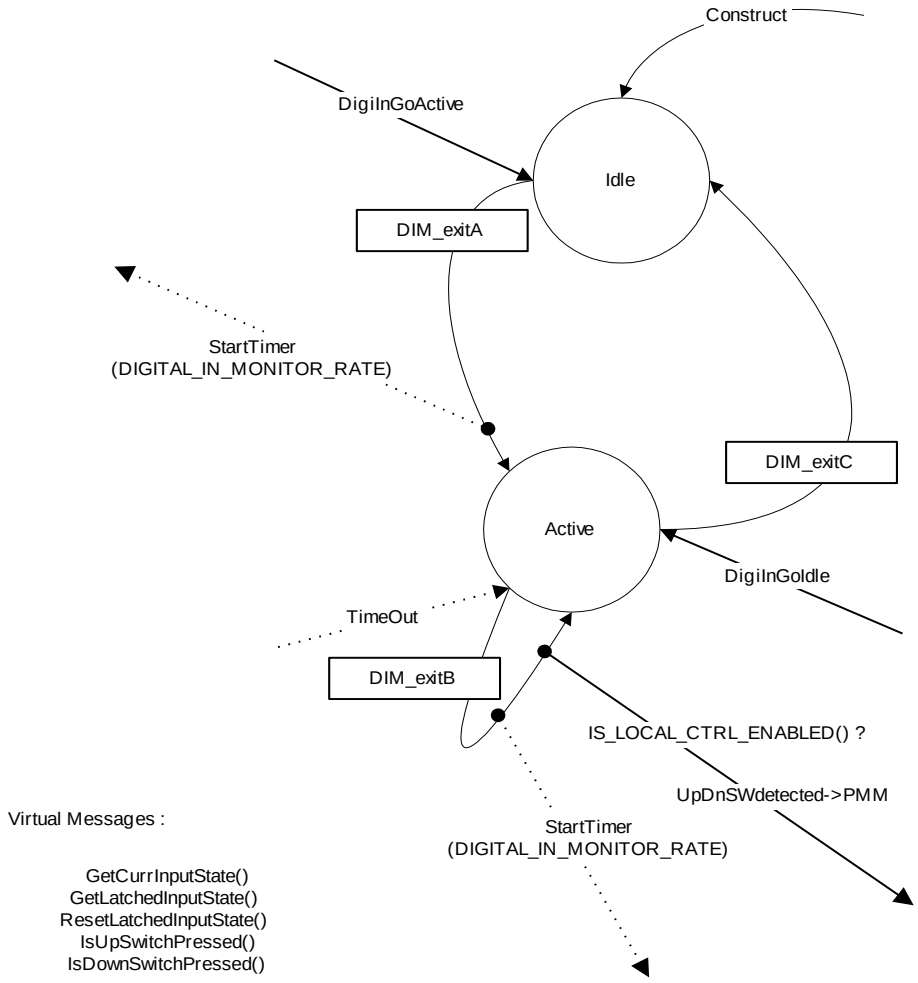


DIM provides one special message for PMM. This is the UpDnSwDetected message. This message is only sent when one of the Up/Down switch has changed state from OFF(switch normal) to ON (switch depressed).

2.7.2 DIM State Transitions

DIM is a periodic task which executes at a given rate. While in ACTIVE, each of the inputs are scanned for its instantaneous state (ON/OFF). The state condition for the individual is latched after 2 successive scans are found to be the same.

The status of the inputs being monitored can be queried through the use of the virtual function services. One of these virtual function is the GetLatchedInputState. DIM provides a service where it latches the input state of any one of the 3 general input is when it is detected to be ON.



IMPORTANT :

An input state stays the same for at least 90 msec. Remember it takes 2 successive scans to be the same before the input changes are considered. This behavior is important since the Up Down Switch Monitor Machine depends on it.

2.8 Up Down Input Monitor Machine (UDM)

Description :

A separate State Machine monitors the Up and Down input status.

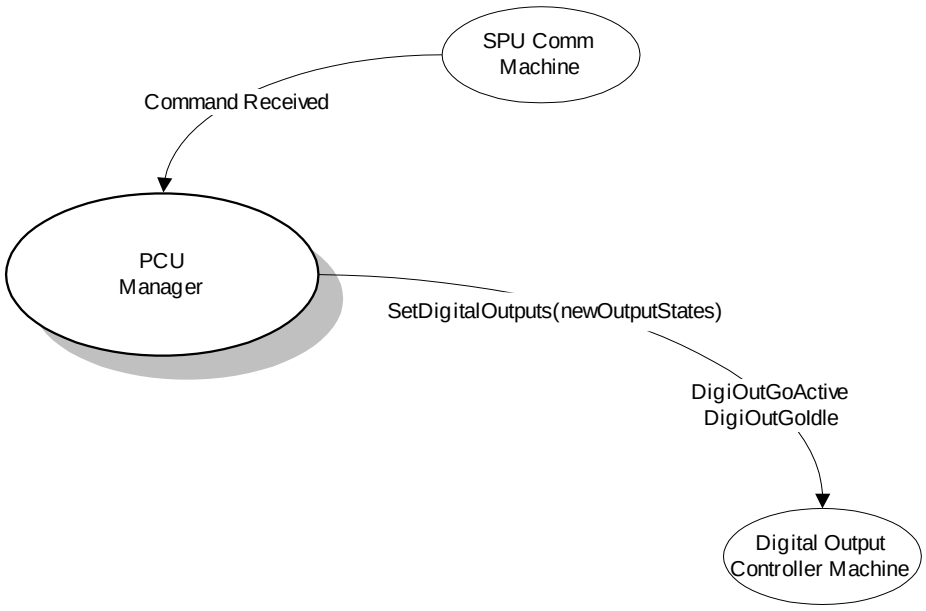
2.9 Digital Output Controller Machine (DOC)

Description :

The PCU provides 3 general Digital Outputs. The Digital Output Controller Machine is in charge of updating the status of these 3 outputs.

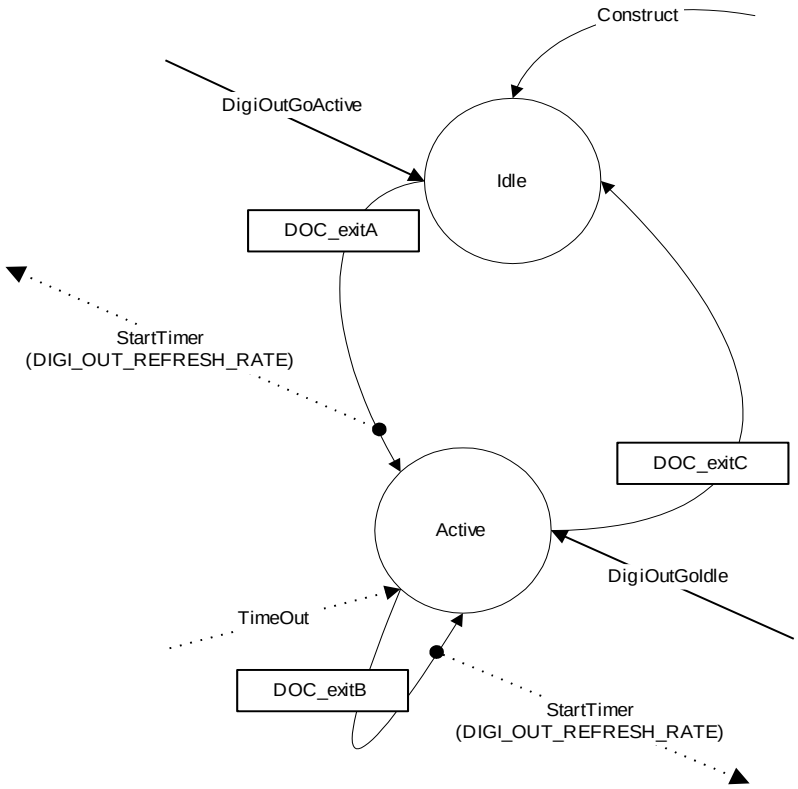
2.9.1 DOC System Interaction

Below is a diagram showing how DOC interacts with the rest of the system :



2.9.2 DOC State Transitions

DOC is a periodic task which executes at a given rate. The states of each of the 3 outputs are refreshed while in the ACTIVE state. Setting the states of the 3 outputs is achieved through the use of the virtual function service.



Virtual Message:

SetDigitalOutputs(newOutputStates)

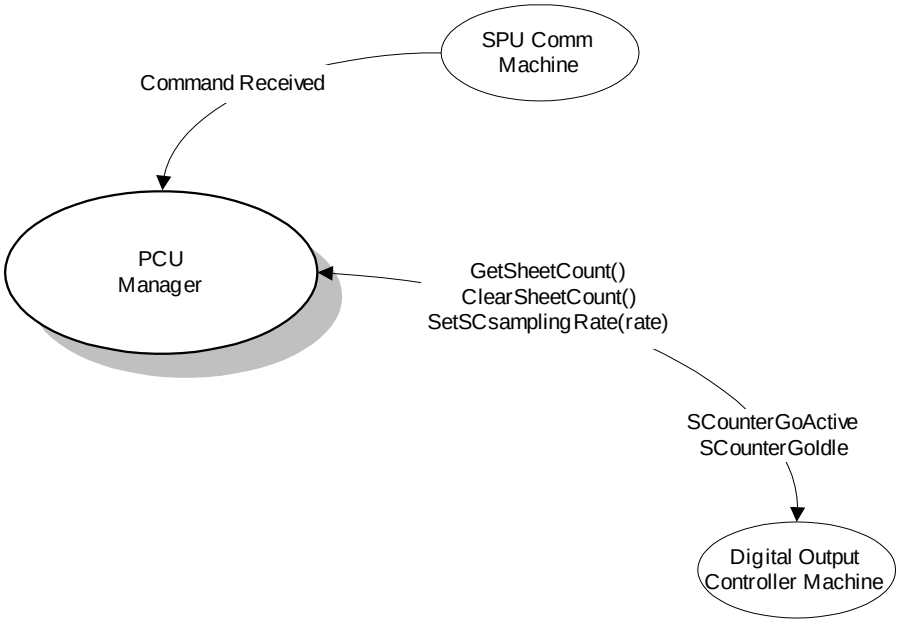
2.10 Sheet Counter Machine (SCM)

Description :

The PCU provides a means of monitoring a sheet counter.

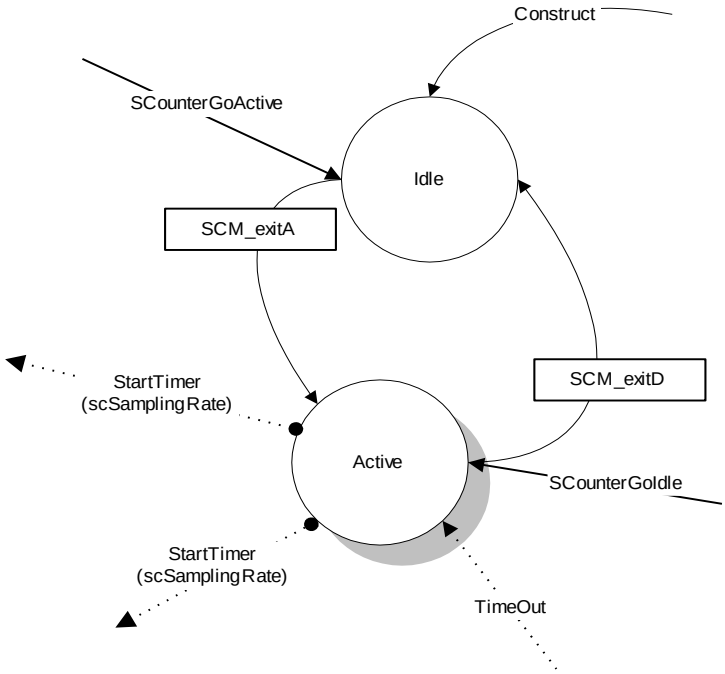
2.10.1 SCM System Interaction

Below is a diagram showing how SCM interacts with the rest of the system :



2.10.2 SCM State Transitions

SCM is simply a pulse counter. It increments its counter register each time a complete pulse is detected. The “sheet count” can be requested through the use of the virtual function service - GetSheetCount.



Virtual Message:

GetSheetCount()
ClearSheetCount()
SetSCsamplingRate()

The Sheet Counter’s Sampling Rate should be adjusted based on the fastest pulse duration that the Sheet Counter input is expected to monitor. Adjusting this rate is part of the SPU command set. See on page .

IMPORTANT :

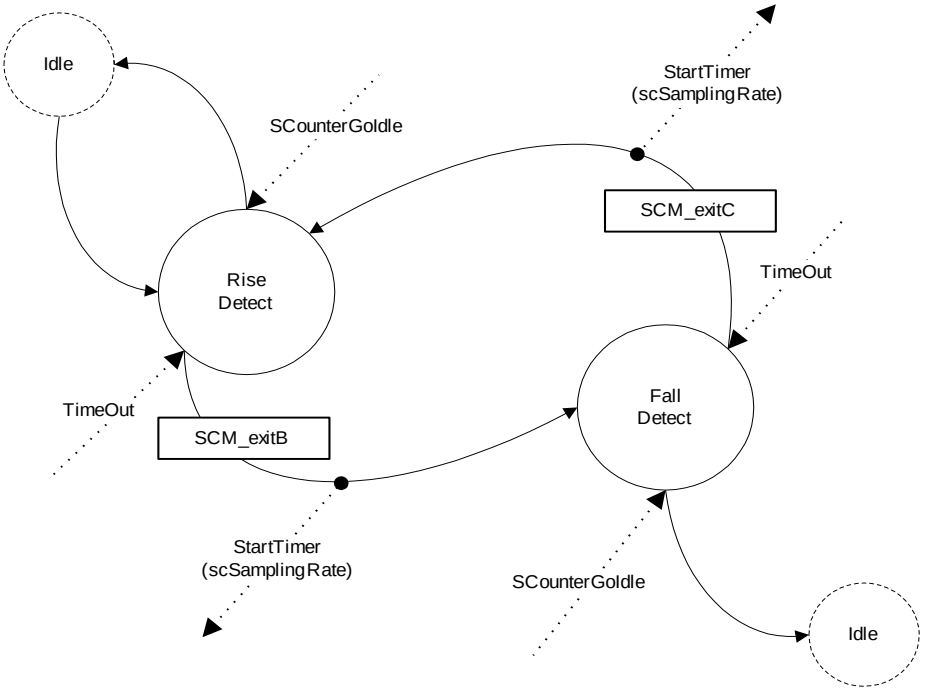
In order for SCM to properly keep track of the sheet counts, the ClearSheetCount virtual message has been provided. The ClearSheetCount message indicates to SCM that the previous request for sheet count has been processed. This is similar to an acknowledge (ACK) signal in order for SCM to maintain counting sheets in the middle of Sheet Count requests. The table below shows this behavior.

Step	Running Sheet Count	Sheet Count Difference	Virtual Function Call (random time)	What happens
0	0			
1	200	200	GetSheetCount	The current sheet count of 200 is retrieved
2	500	300	GetSheetCount	The current sheet count of 500 is retrieved
3	800	300	ClearSheetCount	The sheet count is reset. But since the previous requested sheet count was 500, the sheet count is reset to 300 (800-500)
4	1200	400	GetSheetCount	The current sheet count of 700 (300+400) is retrieved
5	1700	500	ClearSheetCount	The sheet count is reset. But since the previous requested sheet count was 700, the sheet count is reset to 500 (1700-1200)
6	1900	200	ClearSheetCount	No Effect. Did not precede with GetSheetCount
7	2700	800	GetSheetCount	The current sheet count of 1000 (200+800) is retrieved

2.10.2.1 SCM State Transitions, Active

SCM is a periodic task which executes at a given rate. SCM simply monitors the state of the Sheet Count input to determine the occurrence of the sheet count pulse. A complete pulse is considered once a Rising Edge and Falling Edge is detected.

Below is an expanded view of SCM states while it is in ACTIVE :



Note :

A GoIdle command can be processed at any point while SCM is in the ACTIVE state.

2.10.3 Status LED Machine (SLM)

The SLM has been provide to take care of all the Status LED operations.

The Status LED allows the PCU to show its current status be during servicing. The Status LED provides 5 different status indication.

2.10.3.1 Status LED Cadence Definition

A complete cadence is 32 cycles long. Each cycle is latched to the LED (port) at a given interval.

Below is a definition of the different cadences that SLM can provide.

	LED off
	LED on

Scale : 40 msec

Not Powered

1															16
17															32

STEADY : Powered / Not Initialized

1															16
17															32

SLOW PULSE : Powered / Initialized

1															16
17															32

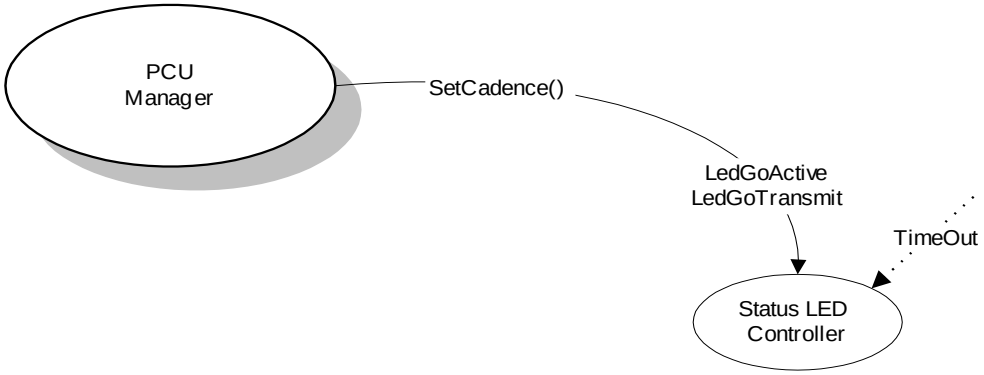
FAST PULSE : Powered / Sending Message to SPU

1															16
17															32

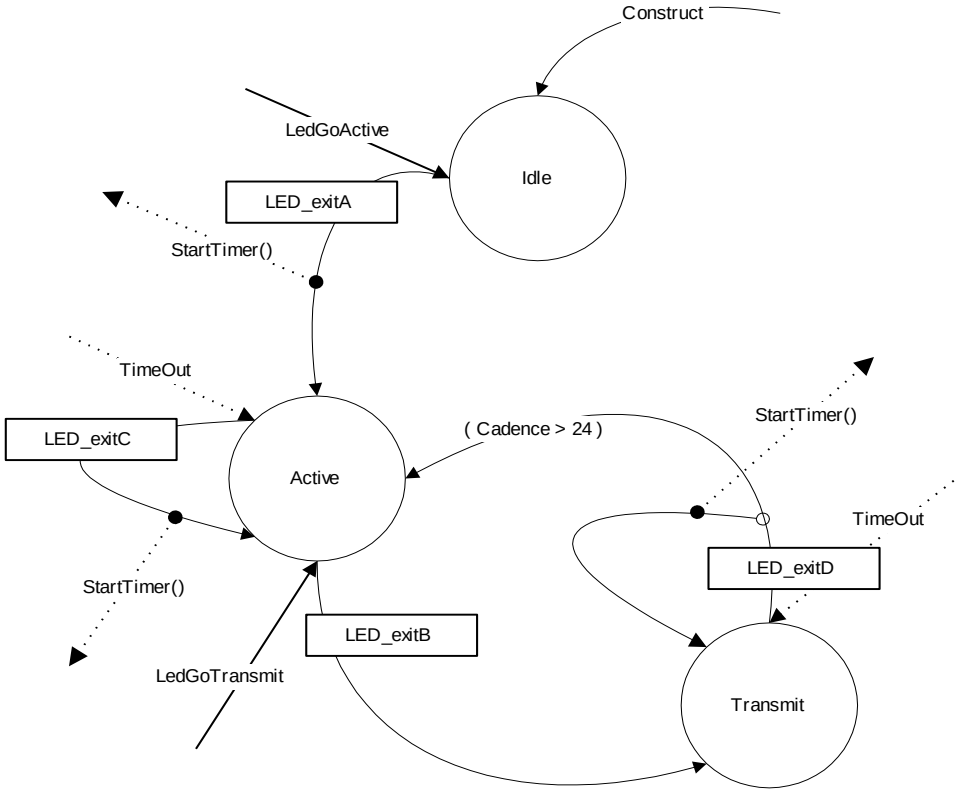
STROBE : Powered / Error Detected

1															16
17															32

2.10.3.2 SLM System Interaction



2.10.3.3 SLM State Transition



Virtual Message:

SetCadence(newCadence)

2.10.4 Timer Service

The Output Compare (OC) function on the 6805C8 is used to provide a 1 millisecond basic timing service. An Interrupt Service Routine is in charge of updating the timer tick count and restarting the OC timer.

Error +/- n

3. Communication Protocol

3.1 General Protocol Format

The PCU 100 employs the hardware UART features of the 6805C8A MCU to perform all of its communication tasks with the SPU. The UART is interrupt driven which allows it to work well within the Kernel Based System.

The servos uses an Asynchronous Transmission method of communication where one character at a time is transmitted or received.

The servo system utilizes a standard RS232 protocol / format of :

1 Start Bit	8 Data Bits	1 Stop Bit

As a convention, the start bit is signified by a voltage level of 0 (Logic 0). A data of 1 is represented by a Logic 1 etc. The stop bit is normally signified by a Logic 1.

With the servo system, this convention is not followed as all logic are reversed.

Start Bit	:	Logic 1
Data Logic 1	:	Logic 0
Data Logic 0	:	Logic 1
Stop Bit	:	Logic 0

The communication voltage levels on the Servo String Bus are GMI-specific and are not RS232 compatible.

3.2 Baud Rate Configuration

This procedure measures the time that the Config Line stays high and uses it as a reference to produce the desired communications speed. This type of speed calibration was implemented for 2 reasons

1. Some servos within the string can only communicate at 4600 BAUD.
2. Due to price sensitivity, the servos do not use a crystal for operation.

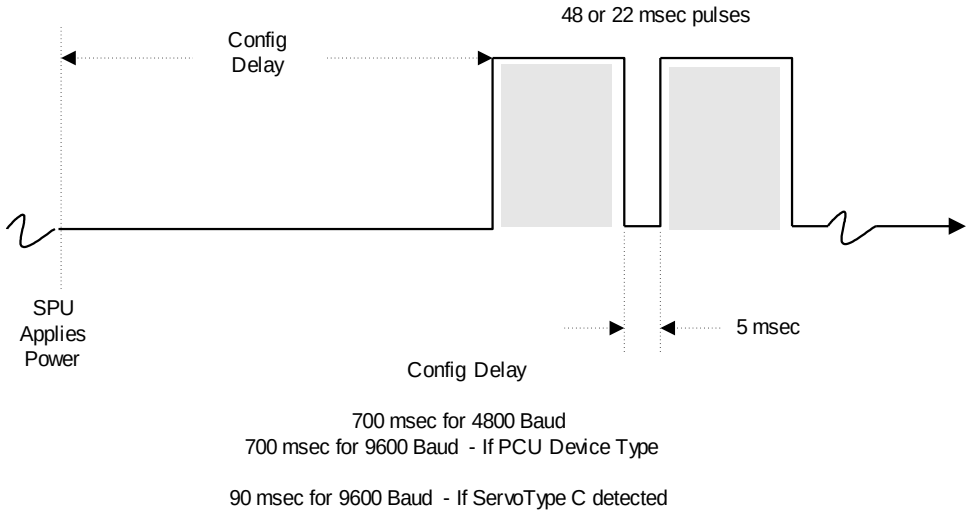
The SPU sends the following pulse:

48 msec = 4800BAUD
22 msec = 9600BAUD

The diagram below shows the Baud Rate Configuration procedure :

The diagram below shows when the Configuration Timing starts:

Figure 1



Note :

The PCU is designed to communicate at both of these specified Baud Rates.

3.3 Communication Protocol Timing Specification

Due to the polling nature of the servo micro-controller units, several delays are implemented within a message packet. This allows the servos to perform system tasks and still manage to communicate with the SPU.

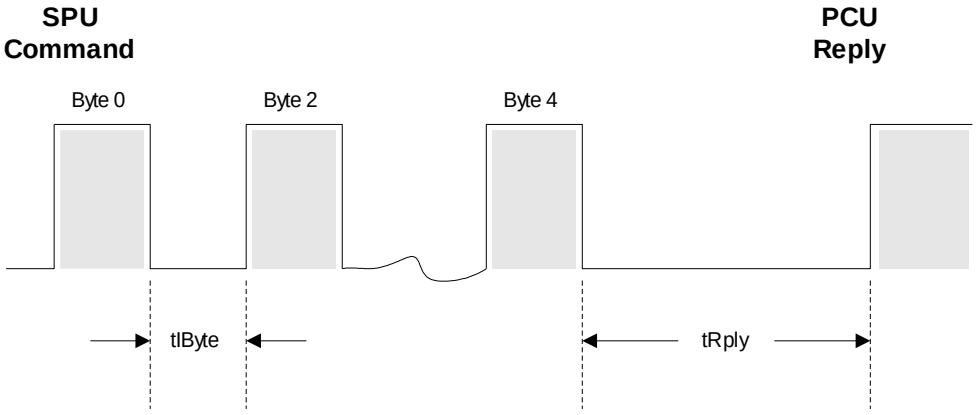
The diagram below shows the current timing implementation of the communications between the SPU and the servo micro-controller units.

Important :

The PCU is designed to be compatible with the old servo's timing specification.

There is no standard or requirements written regarding the timing of these protocols. The diagrams below are timing measurements made on the current servo system, verified using the oscilloscope.

3.3.1 SPU Protocol Timing



Label	approx. Time in milliseconds
tIByte	2.0
tRply	14.0

Note :

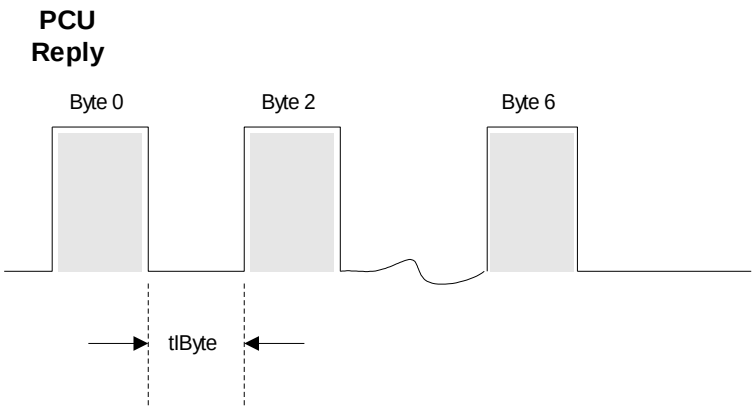
tRply of the PCU is 8 msec slower compared to the Old Servos.

3.3.2 Servo Protocol Timing

The reply of the servo units also contain some delay between data bytes (inter-Byte). This was probably done to compensate for older versions of the SPU which needed these delays.

Note that the inter-byte delay of the servo units defaults to about 2.5 milliseconds. In the old servo this parameter is programmable and can be changed by the SPU.

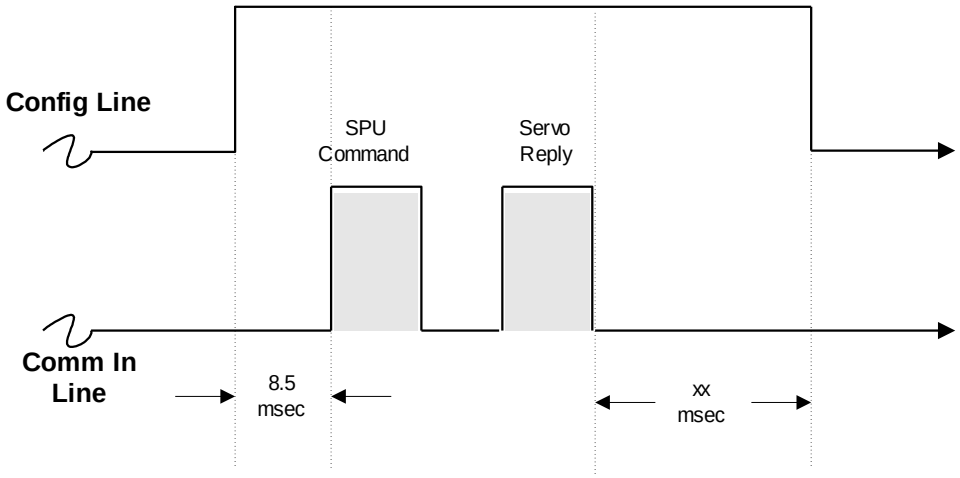
This ability to change the inter-byte delay from the SPU has be taken out. Instead, the inter-byte delay is now a constant for the PCU.



Label	approx. Time in milliseconds
tIByte	2.0

3.3.3 Config Timing

As stated earlier, the Config line serves a special purpose to the servo units. The Config line is used to indicate the start and the end of a transaction on the Servo String Bus. The diagram below shows timing relationship between the Config Line and the SPU command.



Note :

The PCU also uses the Config Line signal to qualify the start of the incoming message packet.

4. Servo Commands/Messages

This section contains commands that can be sent from the SPU :

FULL - This term is used to indicate that a given message is fully supported. The SPU can use the exact same message for the Old Servo Micro-controller as well as the PCU 100 module.

Note :

Some Commands that uses the same Command ID may have data parameters that have a totally different meaning. Since the SPU knows what kind of device it is communicating with, it will adjust the contents of the message packet accordingly.

Command 0x4B - Set Moving Timeout, is one example. The Command ID is exactly the same used by the old servos. However Data 1 is interpreted differently. See page 62.

IMPORTANT :

Not all of the Old Servo commands are required for normal servo operations. Please refer to the Servo Documentation for a list of these used commands.

4.1.1 Command 0x41 - OK Status

From SPU :

Cmd
0x41

PCU Id
0 - 0xFF

Coming from the SPU unit : This is considered a special configuration command and is normally followed by the PCU Id which sets the initial ID of a given unit.

Another form of this command is in a full packet format :

Len	PCU Id	Cmd	CRC 1	CRC 2
0x05	0xID	0x41	CRC 1	CRC 2

In this format the SPU is requesting to check the presence of a given Servo Unit. The Servo Unit with the matching ID replies with a 1 Byte message as shown below.

From Servo Unit :

Cmd
0x41

Compatibility: **FULL**

Description :

Coming from the servo unit : This is a special 1 Byte message (No Header, CRC etc.). The Servo Unit simply returns a 0x41 to the SPU indicating its presence within the string.

4.1.2 Command 0x42 - Set Unit Number

Description :

This command allows the SPU to give a new PCU Id to a Servo Unit that has already been configured. The initial Unit Number (ID) is configured during the initialization or configuration phase of the Servo Units.

From SPU :

Len	PCU Id	Cmd	Data 1	Data 2	CRC 1	CRC 2
0x07	Old ID	0x42	New ID	XxXX	CRC1	CRC2

Note : XxXX - Signifies Don't Care .

Compatibility: **FULL**

4.1.3 Command 0x43 - Close Configuration Line

Description :

This command gives the SPU the ability to set the ConfigLine of a given servo unit so that the other units within the string can start to communicate. A closed line allows the next servo within the string to communicate.

From SPU :

Len	PCU Id	Cmd	CRC 1	CRC 2
0x05	0xID	0x43	CRC 1	CRC 2

Compatibility: **FULL**

4.1.4 Command 0x44 - Open Configuration Line

Description :

This command gives the SPU the ability to clear the ConfigLine of a given servo unit so that the other units within the string can start to communicate. An open line indicates that the next servo within the string can not communicate.

From SPU :

Len	PCU Id	Cmd	CRC 1	CRC 2
0x07	0xID	0x44	CRC 1	CRC 2

Compatibility: **FULL**

Note : The servo units themselves control the ConfigLine most of the time.

4.1.5 Command 0x48 - Get Error Byte

From SPU :

Len	Servo ID	Cmd	CRC 1	CRC 2
0x05	0xID	0x48	CRC 1	CRC 2

From Servo Units :

Len	Servo ID	Status	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	**STATUS**	**XxXX**	**ERROR REG**	CRC 1	CRC 2

Description :

This is a command that initiates the Servo Units to send the 7-Byte reply which contains the error codes. The SPU primarily uses this message to check the last servo in the string during the Config process. The SPU may not even care about the information contained in Data1 and Data2 at this point.

In the old servo operation this message was strictly used for checking the last servo of the string. This functionality has been changed to support the PCU operation. The ERROR REG now contains valid data which actually gives information regarding errors that occur in the PCU operation. See the tables below for more detail.

PULSE COUNT - This data is unused by both the SPU and the PCU.

STATUS - Contains individual bit flags to indicate system status of the Servo Unit

The table below shows how the SPU actually uses the **STATUS** field that is sent by the reply to the following :

1. a reply to a command 0x48
2. or a reply to a command 0x54 (Get Position (AtoD Reading):

Bit No	Mask	Bit Status Description	
00	0x01	*; undefined*	*available*
01	0x02	*; undefined*	*available*
02	0x04	; Motor Move Done	When SET (1), this bit field indicates that the servo motor has reached its target position
03	0x08	*; undefined*	*available*
04	0x10	*; undefined*	*available*
05	0x20	*; undefined*	*available*
06	0x40	*; undefined*	*available*
07	0x80	*; undefined*	*available*

ERROR REG - Contains data indicating any occurrence of errors.

This byte register as sent by the reply to a command 0x48, is **not used by the SPU at all**. However it may serve as a useful debugging tool in the future.

Mask	Error Name	Description
0x01	; ERROR IN CRC WHEN RECEIVING COMMAND	When SET (1), this bit field indicates that a CRC error was encountered in the last packet reception.
0x02	; COMMAND WAS OUT OF RANGE	When SET, this bit field indicates that the last command byte received is out of range or unknown.
0x04	; COUNT TOO LARGE IN MESSAGE	When SET, this bit field indicates that Length (or the Header of the packet) is not within the expected range.
0x08	; BIT BERRST DID EMERGENCY STOP (0x03 - bit position, as seen inside the firmware)	When SET, this bit field indicates that an emergency stop was executed by the servo. Note : How the emergency stop is initiated is unknown. This bit field is **unused** by both the PCU and the SPU.

Mask	Error Name	Description
0x10	; MOTOR AT SW LIMITS	When SET, this bit field indicates a **stall condition** where the target position requested by the SPU is outside of the motor's Software Limit settings.
0x20	; TIMEDOUT WAITING FOR PULSES ; Motor Stuck	When SET, this bit field indicates a **stall condition** where the motor has not reached its position yet and position sensor have stopped moving for a given amount of time. (NO LIMITS HIT)
0x40	; MOTOR AT HW LIMITS	When SET, this bit field indicates a **stall condition** where the motor has hit one of the Min/Max Hardware Limits.
0x80	; SHEET COUNT OVERFLOW	When SET, this bit field indicates that the sheet count has overflowed. The GetSheetCount (command 0x5F) was not executed in time before the rollover occured.

4.1.6 Command 0x49 - Respond with D in Message

Description :

This is a request message from the SPU asking the Servo Units their type. The Old Servo Micro-controller would send a “C” or 0x43. The PCU 100 will send a “D” or 0x44. This is necessary in order for the SPU to differentiate the type of devices that are connected within the string..

From SPU :

Len	PCU Id	Cmd	CRC 1	CRC 2
0x05	0xID	0x49	CRC 1	CRC 2

From PCU :

Cmd
0x44

This is a special 1 Byte reply. These Servo Units respond with an ASCII - “D” (0x44) which denotes a PCU 100 type device.

4.1.7 Command 0x4B - Set Moving Timeout

Description :

This command was originally called “Set Timeout for Pulses”. This command sets the timeout parameter used for determining if the motor is still moving. If the target has not been reached and the motor has not moved within a given Timeout, motor power is disabled and a stall condition is flagged.

From SPU :

Len	PCU Id	Cmd	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	0x4B	**MTO**	XxXX	CRC 1	CRC 2

Compatibility: **Supported, Different Timing Parameter**

MTO

Tick Count	:	3 msec	
Default Value	:	120 ticks	(360 msec)
Range	:	0 - 255 ticks	(0 - 765 msec)

4.1.8 Command 0x4E - Clear Status and Errors

Description :

This command simply allows the SPU to clear all errors latched by the Error Status Byte. This message is generally sent by the SPU after it detects an error from a message reply of a Servo Unit.

Note : This command also clears the Digital Input Latch Register is also cleared. See

From SPU :

Len	PCU Id	Cmd	CRC 1	CRC 2
0x05	0xID	0x4E	CRC 1	CRC 2

Status Byte Definition :

Compatibility: **Supported, The PCU has a different error register definition**

4.1.9 Command 0x51 - Set Coast Wait Time

Description :

After the target has been reached, motor power is disabled and the motor goes into a coasting state. The Coast Wait Time is used to determine if the motor has stopped its coast. This command allows the SPU to set the coast wait time parameter.

From SPU

Len	PCU Id	Cmd	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	0x51	**CTO**	XxXX	CRC 1	CRC 2

Compatibility: **Supported, Different Timing Parameter**

CTO

Tick Count : 3 msec

Default Value : 50 ticks (150 msec)

Range : 0 - 255 ticks (0 - 765 msec)

5. PCU Commands/Messages

The following sections describes the new messages that were specifically designed for the PCU module.

It was intentional to designate new command numbers for the PCU instead of reusing the commands from the old servo. This will be useful for debugging as well.

5.1.1 Command 0x52 - Get Revision

From SPU :

Description :

This is request for the PCU to return the Software Revision Number.

Len	PCU Id	Cmd	CRC 1	CRC 2
0x05	0xID	0x52	CRC 1	CRC 2

From PCU

Len	PCU Id	Stat	Data1	Data2	CRC 1	CRC 2
0x07	0xID	XxXX	**REV HI**	**REV LO**	CRC 1	CRC 2

Below is a table showing the definition of REV HI (Major Rev No) and REV LO (Minor Rev No) :

Areas that are shaded indicate that they are unused.

REV HI								**REV LO**							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	0-99								0-99						

The SPU will interpret this data as follows : Revision " XX . XX ".

5.1.2 Command 0x53 - Move to Target Position

Important : The motor is **inactive** until the position limits are set. See and .

The motor will not move if the target position is over the limits.

Description :

The SPU can send this command while the system is in the “Major Move” positioning stages. The motor will begin to move until the target is reached. The motor then coasts to a stop. Position overshoots are calculated by the SPU and are considered when target value is being set.

To achieve precise positioning, the SPU should send nudges when it detects that the target is close. See.

From SPU

Len	PCU Id	Cmd	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	0x53	**TPOS1**	**TPOS2**	CRC 1	CRC 2

TPOS2 contains the Least Significant Byte while TPOS1 contains the Most Significant Byte. Both TPOS1 and TPOS2 form the 12 bit value of the A/D reading.

Below is a table showing an expanded view of TPOS1 and TPOS2.

Areas that are shaded indicate that it is unused and are masked by the PCU.

TPOS1								**TPOS2**							
				11	10	9	8	7	6	5	4	3	2	1	0
mask 0x0F								mask 0xFF							

Range : 0x0000 - 0x0FFF

Notes :

TPOS 1 and 2 are absolute positions. The direction at which the motor will move is decided by the PCU.

5.1.3 Command 0x54 - Get AtoD Reading

Description :

When the PCU is connected to a potentiometer for analog feedback this command is considered as “Get PCU Absolute Position.”

This command requests the PCU to send back its absolute 12-bit A/D reading. This is considered the current the position. This position reading may change if to the “press operator” decides to change the position manually. This command allows the SPU to react accordingly to any positional changes made by the operator.

From SPU

Len	PCU Id	Cmd	CRC 1	CRC 2
0x05	0xID	0x54	CRC 1	CRC 2

From PCU

Len	PCU Id	Stat	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	**STATUS**	**D1**	**D2**	CRC 1	CRC 2

See on page for a description of the STATUS byte.

D2 contains the Least Significant Byte while D1 contains the Most Significant Byte. Both D1 and D2 form the 12 bit value of the A/D reading (motor position).

Below is a table showing an expanded view of D1 and D2.
Areas that are shaded indicate that it is unused.

D1								**D2**							
15	14			11	10	9	8	7	6	5	4	3	2	1	0

Bit 15 Contains the Direction of last move.
0 = forward
1 = reverse

Bit 14 Indicates if the motor has been moved locally
0 = last move was by SPU
1 = last move was by local control

Note :

A. The SPU should give the local positioning control a higher priority against the console when there is a contention between the two. If the Console detects a local position change event, then it should update to the current position value.
B. With a given interval, the SPU should poll the PCU for its current position and update the values of the console to display any changes.

5.1.4 Command 0x55 - Set Remote Nudge Parameters

Description :

A nudge is a pulse of power that is applied to the servo motor. Different types of motor may require different pulse widths to create a nudge move. This command allows the SPU to set the nudge On-Time parameter.

From SPU

Len	PCU Id	Cmd	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	0x55	**NGOT**	**DIST**	CRC 1	CRC 2

NGOT :

This parameter determines the size of the nudge pulses.

Default : 4 msec

Range : 0 - 255

Units : msec

DIST :

This parameter determines the distance at which the motor will perform the inward nudging as the final positional move.

Default : 70 AtoD counts

Range : 0 - 255

Units : AtoD counts

5.1.5 Command 0x56 - Nudge to Target Position

Description :

This command applies on nudge at a time until the target is reached. The SPU can adjust the Nudge On-Time before sending this command to achieve precise positioning.

From SPU

Len	PCU Id	Cmd	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	0x56	**NPOS1**	**NPOS2**	CRC 1	CRC 2

NPOS2 contains the Least Significant Byte while NPOS1 contains the Most Significant Byte. Both NPOS1 and NPOS2 form the 12 bit value of the A/D reading.

Below is a table showing an expanded view of NPOS1 and NPOS2.

Areas that are shaded indicate that it is unused and are masked by the PCU.

NPOS1								**NPOS2**							
				11	10	9	8	7	6	5	4	3	2	1	0
mask 0x0F								mask 0xFF							

5.1.6 Command 0x57 - Nudge X Times

Description :

Using the current Nudge On-Time parameter, this command will make the PCU apply **X** number of pulses to the motor in a given direction.

This is another means for the SPU to position the servo. The SPU can perform a series of “Nudge and Get Position” until the desired position is reached.

From SPU

Len	PCU Id	Cmd	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	0x57	**NCNT**	**DIR**	CRC 1	CRC 2

NCNT - contains the number of times the nudge pulse will be applied to the motor.

Range - 0 - 255

DIR - indicates the direction at which the motor nudge pulses will be applied :

0x00 - Reverse Direction
0x01 - Forward Direction

Important Note :

The PCU does will not itself check for stall conditions during the nudging process. Stall conditions can be detected by the SPU by reading the positional changes. See

5.1.7 Command 0x58 - Set Local Nudge Factors

Description :

IMPORTANT : The local positioning controls are **inactive** until this command is issued.

This command allows the setting of the following parameters :

1. **Nudge Starting On-Time**
2. **Nudge Acceleration Factor**
3. **Nudge Ending On-Time**

These parameters are used by the SPU to determine how to respond to the Up/Down local motor control switches. This includes the size of one nudge or one button depression, and the speed at which the nudge increases as the button is continuously scanned in the depressed position.

From SPU

Len	PCU Id	Cmd	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	0x58	**NF1**	**NF2**	CRC 1	CRC 2

NF1 contains the data for setting the first 2 parameters, Nudge Starting On-Time and Nudge Acceleration Factor. The table below shows an expanded view of NF1:

NF1							
7	6	5	4	3	2	1	0
High Nibble (Acceleration Factor)				**Low Nibble (Starting On Time)**			
A4	A3	A2	A1	S4	S3	S2	S1

Starting On Time : Default : 4
Range : 0 - 128

Acceleration Factor Default : 2
Range : 0 - 128

NF2 contains the data for the Ending On-Time (1 Byte) :

Ending On-Time : Default : 55
Range : 0 - 255

Note : All units are in milliseconds.

5.1.8 Command 0x59 - Get Digital I/O Status

Description :

The PCU provides 3 digital input monitoring. These inputs have both a “Current State” as well as a “Latched State” register. Whenever a current input state is detected to be set, that value is latched. The SPU can send a command to clear the latched values. See .

This command is a request for the PCU to send back the status of its 3 input ports.

From SPU

Len	PCU Id	Cmd	CRC 1	CRC 2
0x05	0xID	0x59	CRC 1	CRC 2

From PCU

Len	PCU Id	Status	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	XxXX	**INPUT DATA**	**OUT DATA**	CRC 1	CRC 2

Below is a table showing an expanded view of the INPUT DATA and OUT DATA.

Areas that are shaded indicate that it is unused and are masked by the PCU.

INPUT DATA							
7	6	5	4	3	2	1	0
High Nibble (Latched State)				**Low Nibble (Current State)**			
	Inp3	Inp2	Inp1		Inp3	Inp2	Inp1

After the SPU sends a Set Digital Output command (0x5A), it can immediately request for the I/O status to verify that the Set command was correctly executed by the PCU device.

OUT DATA							
7	6	5	4	3	2	1	0
High Nibble (Latched State)				**Low Nibble (Current State)**			
					Out3	Out2	Out1

5.1.9 Command 0x5A - Set Digital Outputs

Description :

The PCU provides 3 digital output control. This command is a request for the PCU to activate or deactivate any of its 3 digital outputs.

From SPU

Len	PCU Id	Cmd	Data 1	Data 2	CRC 1	CRC 2
0x07	0Xid	0x5A	**OUT DATA**	XxXX	CRC 1	CRC 2

The table below shows which bits control which Digital Output ports:

Areas that are shaded indicate that it is unused and are masked by the PCU.

OUT DATA							
7	6	5	4	3	2	1	0
					Output 3	Output 2	Output 1

Bit Val	Output State
1	Activated
0	Deactivate

5.1.10 Command 0x5B - Set Min Move Factor

Description :

The Min Move parameter is used by the PCU to determine the minimum position value change before a “positional change” is considered valid, before the motor is considered to be moving.

This command allows the SPU to adjust the Minimum Move parameter.

From SPU

Len	PCU Id	Cmd	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	0x5B	**MM**	XxXX	CRC 1	CRC 2

MM

Default : 40

Range : 4 - 255

Units : AtoD units

Notes :

With 12 Bits of AtoD data, there are 4095 AtoD units.

With a 5 Volt reference, each AtoD units would be 1.22 mv

A value of 4 is only 00.097% of the AtoD’s full range.

A value of 20 is only 0.5% of the AtoD’s full range. This default value implies that the motor’s position should change by at least 0.5% for it to be considered moving

A value of 255 is 6.2% of the AtoD’s full range.

Important :

The minimum setting takes into account the noise that can be induced in the AtoD input be in the AtoD data readings as seen in the lab environment.

A Min Move value that is too low may not allow the motor to position correctly.

5.1.11 Command 0x5C - Set Min Position

IMPORTANT :

A. The motor is **inactive** until the Min and Max Positions are known.
B. The PCU does not provide range checking for these parameters.
C. Make sure Min < Max and Min >= 0x0000

Description :

The PCU has the capability of being controlled manually by the operator. This command allows the SPU to adjust the minimum position that the PCU will allow the servo to move.

MIN POS2 contains the Least Significant Byte while **MIN POS1** contains the Most Significant Byte. Both **MIN POS1** and **MIN POS2** form the 12 bit value of the A/D reading.

From SPU

Len	PCU Id	Cmd	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	0x5C	**MIN POS1**	**MIN POS2**	CRC 1	CRC 2

Below is a table showing an expanded view of **MIN POS1** and **MIN POS2**.

Areas that are shaded indicate that it is unused and are masked by the PCU.

MIN POS1								**MIN POS2**							
				11	10	9	8	7	6	5	4	3	2	1	0
mask 0x0F								mask 0xFF							

Valid Range : 0x0000 - 0x0FFF

5.1.12 Command 0x5D - Set Max Position

IMPORTANT :

A. The motor is **inactive** until the Min and Max Positions are known.
B. The PCU does not provide range checking for these parameters.
C. Make sure Max > Min and Max <= 0x0FFF

Description :

The PCU has the capability of being controlled manually by the operator. This command allows the SPU to adjust the maximum position that the PCU will allow the servo to move.

MAX POS2 contains the Least Significant Byte while **MAX POS1** contains the Most Significant Byte. Both **MAX POS1** and **MAX POS2** form the 12 bit value of the A/D reading.

From SPU

Len	PCU Id	Cmd	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	0x5D	**MAX POS1**	**MAX POS2**	CRC 1	CRC 2

Below is a table showing an expanded view of **MAX POS1** and **MAX POS2**.

Areas that are shaded indicate that it is unused and are masked by the PCU.

MAX POS1								**MAX POS2**							
				11	10	9	8	7	6	5	4	3	2	1	0
mask 0x0F								mask 0xFF							

Valid Range : 0x0000 - 0x0FFF

5.1.13 Command 0x5E - Get Sheet Count

Description :

This command requests the PCU to send back the current number of pulses logged by the sheet counter of the PCU.

Sheet counters that provides 25 sheets per/second has a cycle of 40 milliseconds. This command needs to be executed every (65535x40 msec) to avoid overflow. The SPU is in charge of clearing the counter.

From SPU

Len	PCU Id	Cmd	CRC 1	CRC 2
0x05	0xID	0x5E	CRC 1	CRC 2

From PCU

Len	PCU Id	Status	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	**XxXX**	**SC1**	**SC2**	CRC 1	CRC 2

SC1 and SC2 form the 16 Bits of sheet count data.

SC2 contains the Least Significant Byte.
SC1 contains the Most Significant Byte.

The SPU should get the sheet count and clear the counter.

The Clear status command normally is executed after the SPU is able to retrieve the sheet count correctly.

Due to the maximum number that the sheet counter can log, the SPU has to retrieve this information before an overflow occurs.

Calculation : Maximum wait :

5.1.14 Command 0x5F - Clear Sheet Count

Description :

This allows the sheet counter to be reset by the SPU.

From SPU :

Len	PCU Id	Cmd	CRC 1	CRC 2
0x05	0xID	0x5F	CRC 1	CRC 2

Note :

As long as the sheet count does not overflow, this command can be sent at a later time after getting the count without loosing sheet count data.

5.1.15 Command 0x60 - Set Sheet Count Scan Rate

Description :

This allows the sheet counter's scanning rate to be controlled. This rate should be dependent on the pulse width of the sheet counter signal.

From SPU

Len	PCU Id	Cmd	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	0x60	**SR**	XxXX	CRC 1	CRC 2

Default : 8 msec
Range : 0 - 255
Units : msec

Note :

Sheet counter signal provides a maximum of 25 sheets per/second.

The signal has a cycle of 40msec. With a 50% duty cycle, the pulse width is 20 msec. To provide at least 2X sampling rate, the scan time = 10 msec. To account for other timing latencies a scan time of 8 milliseconds was chosen.

5.1.16 Command 0x61 - Set PCU Mode

Description :

There are instances where the PCU may want to disable Local Positioning Controls. This command allows the SPU to individually enable or disable sections of the PCU system. This command can also be used to achieve the effect of having a PLC or ANALOG mode setting.

This command can be sent at any time during the PCU's operation.

Also note that It is beneficial to the operation of the PCU system to disable any feature that are not actually being utilized. Timing latencies are improved.

From SPU

Len	PCU Id	Cmd	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	0x61	**PMOD**	XxXX	CRC 1	CRC 2

The table below shows which bits are assigned to the individual features:

Mask	Feature Control
0x01	Local Positioning Controls (see Note 1)
0x02	*undefined*
0x04	*undefined*
0x08	*undefined*
0x10	*undefined*
0x20	*undefined*
0x40	*undefined*
0x80	*undefined*

Bit Val	Feature State
1	Activated
0	Deactivate

Notes :

1. The Local Positioning section and the Digital Input section are interdependent. Enabling the Local Positioning Bit automatically enables the Digital Input section. However the Digital Input section can separately be enabled.
2. The Digital Inputs can not be disabled if the Local Positioning feature is enabled.

3. The Analog system is comprised of the Motor Controller and the AtoD reader. However the AtoD reader can be enabled separately. The AtoD Reader can not be disabled if the Analog System Bit is enabled.

5.1.17 Command 0x62 - Get PCU Mode

Description :

Several sections of the PCU are implicitly enabled by sending certain commands. This command allows the SPU to check the current mode settings of the PCU.

From SPU :

Len	PCU Id	Cmd	CRC 1	CRC 2
0x05	0xID	0x62	CRC 1	CRC 2

From PCU :

Len	PCU Id	Stat	Data 1	Data 2	CRC 1	CRC 2
0x07	0xID	XxXX	**PMOD**	XxXX	CRC 1	CRC 2

5.2 Unsupported Commands

6. Hardware

6.1 CPU

6.2 Port Assignments

The table below shows how each of the ports of the Motorola HC6805 C8 are used.

The PCU 100 uses the 40 pin DIP package.

Pin Number	Pin Name	Usage
1	**RESET/**	**Input, CPU Reset/** Input from VCC monitor circuit
2	**IRQ/**	Input, unused/available IRQ/ Pulled up to VCC for future pulse counting (?)
3	**Vpp**	Supply, Programming Power No Connection
4	**PA7**	**Output, "Digital" Signal 3** Multipurpose output
5	**PA6**	**Output, "Digital" Signal 2** Multipurpose output
6	**PA5**	**Output, "Digital" Signal 1** Multipurpose output
7	**PA4**	**Output, Serial ADC Chip Select/**
8	**PA3**	**Output, SPU Xmit On, Xmit Off/**
9	**PA2**	**Output, SPU Config Pass**
10	**PA1**	**Output, Motor Control: Forward (+)**
11	**PA0**	**Output, Motor Control: Reverse (-)**
12	**PB0**	**Input, Motor Minimum Limit/**
13	**PB1**	**Input, Motor Maximum Limit/**
14	**PB2**	**Input, "v" Down Switch/** Operator manual motor control switch
15	**PB3**	**Input, "^"Up Switch/** Operator manual motor control switch
16	**PB4**	**Input, Clear/** Operator switch to clear a latched output
17	**PB5**	**Input, Press Running/** Qualifier to allow register and some sweep motors to move
18	**PB6**	**Input, Press Impression On/** Qualifier for sheet count input
19	**PB7**	**Input, SPU Config In**
20	**Vss**	**Supply, CPU Ground**

Pin Number	Pin Name	Pin Usage
21	**PC7**	**Output, PCU 6805 Status LED** Pin has high source and sink capability
22	**PC6**	**Output, PCU Busy LED** Operator manual move switch “wait” signal
23	**PC5**	unused & available I/O programmed as output
24	**PC4**	unused & available I/O programmed as output
25	**PC3**	unused & available I/O programmed as output
26	**PC2**	unused & available I/O programmed as output
27	**PC1**	unused & available I/O programmed as output
28	**PC0**	unused & available I/O programmed as output
29	PD0 **RDI**	**Input, SPU Comm Receive**
30	PD1 **TDO**	**Output, SPU Comm Transmit**
31	PD2 **MISO**	**Input, ADC Serial Data In**
32	PD3 **MOSI**	Unused & Reserved Serial Data Out programmed as output
33	PD4 **SCK**	**Output, ADC Serial Data Clock**
34	PD5 **SS/**	Unused & Reserved Slave Select/ programmed as input & pulled up to VCC
35	PD6 **TCMP**	Output, Unused & Reserved Compare Function
36	**PD7**	**Input, Press Sheet Count/** Input only pin
37	**TCAP**	Input, Unused & Reserved Capture Function. Cuttable ground connection
38	**OSC2**	Input, unused crystal circuit no connection
39	**OSC1**	**Input, 4.00 MHz CMOS oscillator**
40	**Vdd**	**Supply, CPU VCC (+5V) Power**

7. Issues