User firmware may be developed in two ways for execution on the Inertial Solutions Processor Card. The best way is to purchase the source code from Inertial Solutions (IS), and purchase the Introl-C compiler from Introl. With this approach all the Application Program Interfaces (API's) are exposed, and the API calling conventions are conveniently obeyed by the compiler (see www.introl.com - User Information – Application Notes).

However, this note describes the alternative, where the user interface to the hardware is through a set of assembly language API's that are stored at vector table index 90 and above (as listed below). While less flexible than the use of the IS source code, this approach may be simpler because learning the existing body of IS software is unnecessary.

Custom user programs are stored in EPROM starting at address F0000. At initialization and before the IS tasker is started, address F0000 is tested for the presence of any program code. If code is present, then a call is issued to the user's initialization code that must be present at address F0000. After the user code has installed function address' into the vector table and executed a return (RTS), the IS code will finish initializing and start the IS tasker. The tasker shares the CPU with those user functions put into the vector table at user program initialization. These are a possible two background user tasks and six preemptive user tasks.

The vector table (starting at system address 0) contains space for the two user backgound task addresses and the six preemptive task addresses. Initially, the reserved slots point to a function that does nothing but return (RTS). By inserting a custom function address, the user can tap into the tasker's execution chain. For example:

```
MOVE.L  #user_function,90*4    ;Replaces the background task default RETURN function.
```

User functions that require execution at specific times may tap into the tasker's preemptive execution chain. Taps are provided for execution every 1/32nd sec, 1/16th sec, 1/8th sec, 1/4th sec, ½ sec, and 1 second. When possible, user work is done in the background tap where all CPU excess capacity is available to the user. However, a conservative recommendation is for the user to limit CPU loading within the preemptive tasks as below:

Task	msec
preempt32	2
preempt16	4
preempt8	8
preempt4	16
preempt2	32
preempt1	64
Background	

Most vector table content is installed at IS firmware initialization thus preserving user interface compatibility between IS firmware builds. The problem of incompatible compiler stack based calling conventions (user compiler verses Introl compiler) is avoided by defining an assembly language vector table calling convention. It uses CPU registers to pass and return data and pointers (D0, D1, A0). Registers D2 – D7, A2 – A6 are preserved. Registers D0, D1, A0, A1, CCR are destroyed.

For example, using the assembly language calling convention, the user's firmware calls the ROM resident API function for controlling a PWM output channel as follows:

```
…
MOVE.L  #1572,D0       ;Example of PWM width passed to API function in D0
MOVE.L  #0001,D1       ;Example of PWM channel# passed to API function in D1
MOVEA   100*4,A1       ;Load the 4 byte PWM function address stored at 0 plus index 100
JSR     (A1)           ;Call the API.
;
```

1. As discussed above, Vector table entries 90 – 97 are available for installation of custom function pointers. Default is a function that does nothing but return. Most of the user's processing is done in one or two of the background

task at index 90 and 91. The preemptive task (index 92 - 97) are also available, however the user may safely leave any task unused.

2. The PWM function at Index 100 allows separate control of pulse width high time for each of the 8 servo output channels. Typical Pulse Width Modulation (PWM) of the servo position varies from a low of 1 msec (D0 = 1024) to center at 1.5 msec (D0 = 1572), to a high of 2 msec (D0 = 2048).
3. The function at Index 101 modifies the PWM period separately for each of the 8 PWM channels. Normally, the 20 msec (D0 = 20,480) default is unmodified. Maximum permissible count is 32k.
4. The function at index 102 reads any one of 4 Input Transition counters at TPU channel 11, 12, 13, or 14.
5. The function at index 103 clears any one of 4 Input Transition counters at TPU channel 11, 12, 13, or 14.
6. The function at index 104 returns a TPU channel RAM address for the requested channel (0 – 15).
7. The function at index 108 returns the interrupt counter of the Periodic Interval Timer (PIT). The counter ticks at approximately 1 msec per tick (1/1024 seconds).
8. The user may install a function pointer at index 109. The function is called once per second after the main GPS message is received, and after the message contents are extracted and stored into a structure.
9. The function at index 110 returns the base address of the GPS structure referred to at 109 above.
10. The function at index 111 returns the base address where the 8 channel 12 bit Analog to Digital conversion results are stored. Each of the 8 channels are over sampled (512 samples per second), thus providing considerable flexibility to how frequently the input data will be averaged. Customer use of the Inertial Solutions source code makes use of the design flexibility possible. However, the default configuration provided here is that the input data pointed to is giving a new averaged input data 32 times per second. For example, the user may have installed a custom function at index 96 (Preempt32) that wishes to use the new ADC input data (as pointed to by a call to this function at index 111).
11. The processor card has three serial UARTs . By default Inertial Solutions (IS) firmware does the character transmission and receive on all three channels through 2048 byte queues. For example, the IS firmware communicates to and from the GPS receiver through the GPSQ. However, the user may disconnect the IS firmware from access to any one or from all three queues' by calling the functions at index 128, 129, or 130 below.
12. User serial I/O may interface to the queues by calling functions at index 112 through 127 below. IS queue access should be disabled first. For example, to receive a character from the GPS queue, the user would first call the function at index 114 to learn how many received characters are waiting if any, then call the function at index 115 to get a character. Putting characters into the queue can be done without checking for space (at index 112) by calling the put function at index 113. However, if the queue is becoming full and the output function must wait (in order to avoid dropping characters on overflow), then the transmitting function should release the CPU while waiting. The CPU will automatically return shortly to resume operation just after the previous point of release. CPU release is done by calling the sleep function at index 134. Only the background task may use the queue interfaces (not preemptive tasks).
13. RF Queue access is identical to GPS Q access described above. Index 116 – 119 serve the RFQ. Input and output is 9600 bps for both queues.
14. TP Queue (Test Port) access is the same as the GPS Q and RF Q above. Index 120 – 123 provides the user access to the Test Port queue from background tasks. Default bit rate is 38.4 kbps. IS firmware uses the Test Port for development and test including a set of pre-defined keyboard commands and character based CRT displays. The user may share by transmitting from the background task into the "put" TP Q without removing IS access. However, the function at index 130 can be used to reserve TPQ and the ITQ access solely to the user.
15. IT Queue access is transmit only. It shares the same transmit UART port as the PutTPQ above, except that "putting" into the queue is from pre-emptive task. Calling the function at index 130 will disconnect the IT Q from use by the IS firmware. When disconnected, the IT Q is normally also unused by the user because there is no point in having two parallel paths to a single UART transmit channel. When IT Q is disconnected the user should use the PutTPQ above (at index 121).
16. The processor board has a yellow and a red LED. The RED LED may be turned on, turned off, or toggled by calling the functions a index 131 – 133. By default the yellow LED is blinking to show the processor board is alive.
17. User background tasks may either return (RTS) or SLEEP (at index 134). SLEEP'ing preserves the local variables while releasing the CPU to other background tasks. Execution is resumed at the first instruction past the call to SLEEP. The user should never put the program in any task into a wait loop, but rather should call the function SLEEP, thus letting the CPU continue doing other work while the user background task is waiting for a condition. SLEEP is for background task only, not the preemptive task. User preemptive task exit always uses a return instruction (RTS) as compilers will do for non-interrupt routines.
18. The function at index #135 will result in an exit to the Motorola CPU32BUG monitor program.
19. The function at index #136 returns the 4 Bit DIP switch input: D0 = SW2, D1 = SW1, D2 = SW3, D3 = SW4

Index	Function	Call With D0	Call With D1	Return D0	Description
90	User Bgnd1				User tap into background task. Sleep on completion to release.
91	User Bgnd2				User tap into background task. Sleep on completion to release.
92	User Preempt1				User tap into tasker. Execute 1 time per second
93	User Preempt2				User tap into tasker. Execute 2 times per second
94	User Preempt4				User tap into tasker. Execute 4 times per second
95	User Preempt8				User tap into tasker. Execute 8 times per second
96	User Preempt16				User tap into tasker. Execute 16 times per second
97	User Preempt32				User tap into tasker. Execute 32 times per second
98	Reserved				
99	Reserved				
100	WRITE_PWM	Count	Chn#		Write D0 count into TPU Chn (0-7) PWM high time (1 usec/count)
101	WRITE_PERIOD	Count	Chn#		Write D0 count into TPU Chn (0-7) PWM period (1 usec/count)
102	READ_ITC		Chn#	Count	Read the TPU Input Transition Counter (Chn 11 - 14).
103	CLEAR_ITC		Chn#		Clear the TPU Input Transition Counter (Chn 11 - 14).
104	TPU_RAM		Chn#	Address	Get TPU Channel Ram starting Address
105	Reserved				
106	Reserved				
107	Reserved				
108	PIT_COUNT			Count	PIT Interrupt count (= 1/1024 Seconds each)
109	User function				Called on new GPS msg 1003 processing complete
110	GPS_DATA			Address	Address of unpacked GPS msg 1003 data structure.
111	DAS_DATA			Address	Address of most current DAS (chn 7,0,1,2,3,4,5,6) sample
112	ctxGPSQ			Count	# Chr in txGPS Q
113	putGPSQ	Byte			Write to txGPS Q
114	crxGPSQ			Count	# Chr in rxGPS Q
115	getGPSQ			Byte	Read fr rxGPS Q
116	ctxRFQ			Count	# Chr in txRF Q
117	putRFQ	Byte			Write to txRF Q
118	crxRFQ			Count	# Chr in rxRF Q
119	getRFQ			Byte	Read fr rxRF Q
120	ctxTPQ			Count	# Chr in txTP Q
121	putTPQ	Byte			Write to txTP Q
122	crxTPQ			Count	# Chr in rxTP Q
123	getTPQ			Byte	Read fr rxTP Q
124	ctxITQ			Count	# Chr in txIT Q
125	putITQ	Byte			Write to txIT Q
126	Reserved				
127	Reserved				
128	swGPSQ	long			Switch GPSQ 0 = IS; 1 = User
129	swRFQ	long			Switch RFQ 0 = IS; 1 = User
130	swTPQ	long			Switch TPQ 0 = IS; 1 = User
131	onred,				On red LED
132	offred				Off red LED
133	chgred				Change red LED
134	SLEEP_BGND				Release the CPU from a background task
135	EXIT				Exit to Motorola CPU32BUG
136	DIPSWITCH			Byte	4 Bit switch input: D0 = SW2, D1 = SW1, D2 = SW3, D3 = SW4