

Practical UML Statecharts in C/C++ +, Second Edition: Event-Driven Programming for Embedded Systems

By Miro Samek

Newnes, 2008

ISBN-10: 0750687061

ISBN-13: 978-0750687065

Companion Website: www.state-machine.com/psicc2

Errata for first printing (October 2008)

1.1 Errors on the Back Cover:

Location	Is	Should be
First Bullet, “Understand State Machine Concepts”	From traditional finite state automated to modern UML state machines	From traditional finite state automata to modern UML state machines

1.2 Errors in the Text:

Location	Is	Should be
Page xxi, 5 th paragraph	Cortex-M3	Cortex-M3
Page xxi, 5 th paragraph	form Luminary Micro	from Luminary Micro
Page xxi, last paragraph	Max OS X	Mac OS X
Page xxi, footnote 2	EKIEV-LM3S811	EKI-LM3S811
Page 2, 1 st paragraph	catalogue	catalog
Page 9, 6 th paragraph	game-ev-lm3s811.eww	game.eww
Page 13, Listing 1.1, explanation section (3)	Section 1.7	Section 1.6
Page 18, Figure 1.4(14)	DESTROYED_MINE (type)	DESTROYED_MINE (score)
Page 15, Listing 1.1 explanation section (17)	(17) The function <code>QF_poolInit()</code> initializes...	(17) The function <code>QF_psInit()</code> initializes...
Page 20, Figure 1.4 explanation section (12)	... Missile posts the <code>MISSILE_IMG(x, y, bmp)</code> event to Table.	... Missile posts the <code>MISSILE_IMG(x, y, bmp)</code> event to Tunnel.
Page 20, Figure 1.4 explanation section (13)	Table renders the Missile bitmap... <code>HIT_MINE(score) ...</code>	Tunnel renders the Missile bitmap... <code>DESTROYED_MINE(score) ...</code>
Page 20, last paragraph	... actions performed by an active object depend as much on the events it receives as on the internal mode of the object.	... actions performed by an active object depend as much on the internal mode of the object as on the events it receives.
Page 22, Figure 1.5(5)	<code>HIT_MINE(score)</code>	<code>DESTROYED_MINE(score)</code>
Page 24, last paragraph	argumentation	reasoning
Page 25, Figure 1.6(12)	<code>QActive_postFIFO(Table, ...</code>	<code>QActive_postFIFO(Tunnel, ...</code>
Page 26, Figure 1.6 explanation section (7)	The <code>PLAYER_TRIGGER</code> internal transition...	The <code>PLAYER_TRIGGER</code> internal transition...
Page 26, Figure 1.6 explanation section (8)	... The score is not posted to the Table at this point.	... The score is not posted to the Tunnel at this point.
Page 30, Figure 1.9 explanation section (7)	The exit action in the “used” state posts the <code>MINE_DISABLED(mine_id)</code> event to	The exit action in the “used” state posts the <code>MINE_DISABLED(mine_id)</code> event to the Tunnel active object... (see also

	<p>the Tunnel active object... (see also Figure 1.9(4))... Note that generating the MINE_DISABLED(mine_id) event in the exit section from "used" ...</p>	<p>Figure 1.7(4))... Note that generating the MINE_DISABLED(mine_id) event in the exit section from "used" ...</p>
<p>Page 31, Fig 1.9</p>	<p>The internal transition TIME_TICK in state "used" is:</p> <pre> TIME_TICK [me->x + GAME_MISSILE_SPEED_X < GAME_SCREEN_WIDTH] / me->x += GAME_MISSILE_SPEED_X; postFIFO(Tunnel, MISSILE_IMG(me->x, me->y, MISSILE_BM P)); </pre>	<p>The internal transition TIME_TICK in state "used" should be:</p> <pre> TIME_TICK [me->x >= GAME_SPEED_X] / me->x -= GAME_SPEED_X; postFIFO(Tunnel, MISSILE_IMG(me->x, me->y, MINE2_BMP)); </pre> <p>(see also state diagram below)</p>

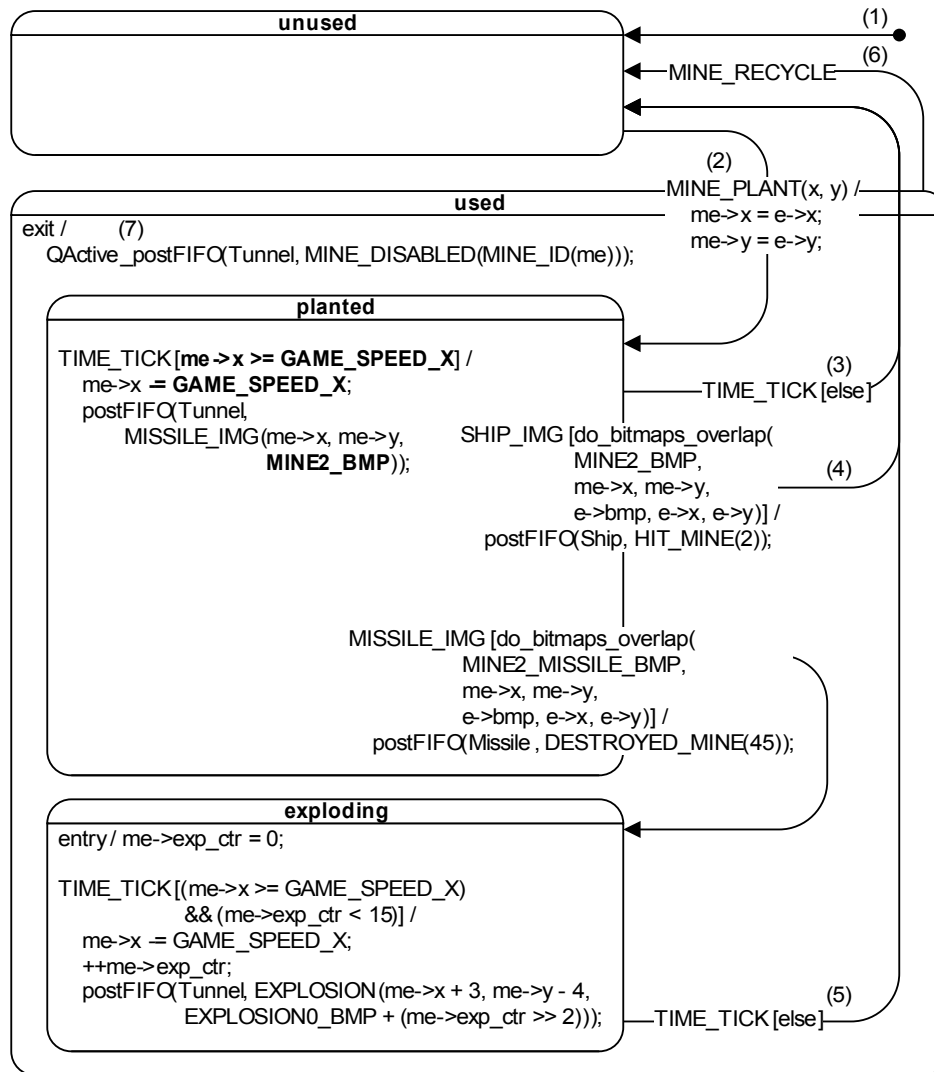


Figure 1.9: Mine2 state machine diagram.

Location	Is	Should be
Page 33, Listing 1.2	<code>/* From Missile the Tunnel ... */</code>	<code>/* From Missile to the Tunnel ... */</code>
Page 46, Listing 1.6 explanation section (9)	<code>return QHandled()</code>	<code>return Q_HANDLED()</code>
Page 47, Listing 1.6 explanation section (16)	<code>return QHandled()</code>	<code>return Q_HANDLED()</code>
Page 53, 4 th paragraph	While the coding ...	While the coding ...
Page 54, last paragraph	built-in	built-in
Page 60, footnote 2	Ignore at this print...	Ignore at this point...

Page 71, end of first paragraph	superstate "hating"	superstate "heating"
Page 88, Fig 2.11	The transition G in state "s21" goes to "s11"	The transition G in state "s21" should go to "s1" (see also state diagram below)

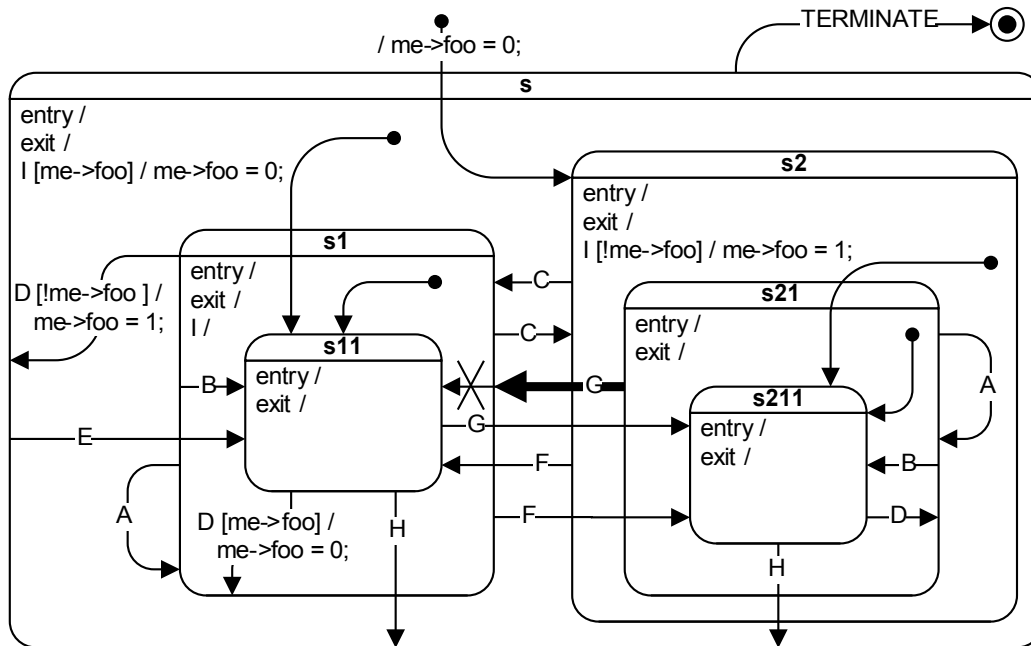


Figure 2.11: Hypothetical state machine that contains all possible state transition topologies up to four levels of state nesting.

Location	Is	Should be
Page 89, first line	...UML state machines...	...UML state machines...
Page 89, 1 st paragraph	...defined in the direct target state "s21."	...defined in the direct target state "s1."
Page 90, 4 th paragraph, 2 nd line	States "s2" and d both ...	States "s2" and "s" both ...
Page 96, Section 2.4.6, end of paragraph	The actual C implementation...	The actual C++ implementation...
Page 91, Section 2.4 title	Designing A UML State Machine	Designing a UML State Machine
Page 93, In TIP	OPER (operand)	OPER (operator)
Page 94, footnote 10	factorize out	factor out

Page 97, Figure 2.18	OPER [e->keld == '-'] (to "negated1") OPER [e->keld == '-'] (to "negated2")	OPER [e->keyld == '-'] OPER [e->keyld == '-']
Page 104, 1 st paragraph, 2 occurrences	tcp101\bomb	tcp101\\bomb
Page 104, 2 nd paragraph	Section 1.1	Section 1.2.1
Page 113, 6 th paragraph	qpc\examples\cortex-m3\dos\iar\game\bsp.c	qpc\examples\cortex-m3\vanilla\iar\game-ev-lm3s811\bsp.c
Page 116, Listing 3.2 explanation section (3)	This typedef defines Tran type as a pointer to the StateTable struct and a pointer to the Event struct as arguments and returns uint8_t. The value returned from the transition function represents the next state for the state machine after executing the transition...	This typedef defines Tran type as a pointer to the StateTable struct and a pointer to the Event struct as arguments and returns void....
Page 121, Listing 3.4 explanation section (17)	The sate table ...	The state table ...
Page 125, Figure 3.6, to immediate right of "Bomb 3" class	state_	state
Page 132, Figure 3.8, to immediate right of the "Bomb" class	state_	state
Page 137, Listing 3.7(7)	(void) (*me->state) (me, &QEP_reservedEvt_[Q_EXIT_SIG])	(void) (*me->state) (me, &QEP_reservedEvt_[Q_ENTRY_SIG])
Page 138, Listing 3.8 explanation section (1)	#include "qp_port.h"	#include "qep_port.h"
Page 141, Listing 4.1 explanation section (14)	Figure 3.1(1)	Figure 3.2(1)
Page 153, Listing 4.1	80x88	80x86
Page 153, Listing 4.1	qep_pkg.h - internal, packet-scope interface	qep_pkg.h - internal, package-scope interface
Page 155, In code snippet	QEP_SIGNAL_SIZE	QP_SIGNAL_SIZE
Page 156, footnote no. 4	Casting from subclass to superclass is called in OOP downcasting...	Casting from superclass to subclass is called in OOP downcasting...

Page 156, last paragraph	became	become
Page 159, Listing 4.4 explanation section (5)	Calc const *	CalcEvt const *
Page 160, Listing 4.4(6)	Q_HANLDED ()	Q_HANDLED ()
Page 160, 5 th paragraph	state "operand1"	state "int1"
Page 168, first NOTE box, fist line	QEQ_EMPTY_SIG	QEP_EMPTY_SIG
Page 169, in code snippet, 2 nd macro	... /* QS software tracing instrumentation for state entry */	... /* QS software tracing instrumentation for state exit */
Page 169, in code snippet 3 rd macro	... /* QS software tracing instrumentation for state exit */	... /* QS software tracing instrumentation for state entry */
Page 170, first paragraph	Myrphy	Murphy
Page 170, Section 4.5.6 Bullet 4.	Execution of the entry actions to the "result" state	Execution of the entry actions to the "ready" state
Page 170, Section 4.5.6 Bullet 5.	Execution of the actions associated with the initial transition defined in the "result" state	Execution of the actions associated with the initial transition defined in the "ready" state
Page 175, Listing 4.11 explanation section (2)	initial transition	transition
Page 183, Listing 4.12 explanation section (10)	(10) The topologies shown in 4.6(G) and (H) require traversal of the target state hierarchy stored in the array path[] to find the match with any of the superstates of the source.	(10) Because every scan for a match with a given superstate of the source exhausts all possible matches for the LCA, the source's superstate can be safely exited.
Page 183, Listing 4.11 explanation section (11)	(11) Because every scan for a match with a given superstate of the source exhausts all possible matches for the LCA, the source's superstate can be safely exited.	(11) The topologies shown in 4.6(G) and (H) require traversal of the target state hierarchy stored in the array path[] to find the match with any of the superstates of the source.
Page 185, last line	<qp>\qp\include\qevent.h	<qp>\qpcpp\include\qevent.h
Page 188, 4 th paragraph	... Listing 4.5 provides two examples of regular state transitions.	... Listing 4.5 provides an example of a regular state transition.
Page 191, 1 st paragraph	Cac1::begin()	Calc1::begin()
Page 196, 2 nd paragraph	event that dispatched the state machine	event that is dispatched to the state machine
Page 197, end of 3 rd	IDC_1_9_SIG	DIGIT_1_9_SIG

paragraph		
Paragraph 209, Listing 5.1 UltimateHook_generic, case C_SIG	generic-C	Generic:C
Page 210, Listing 5.1 explanation section (1)	Every QEP application needs to include qep_porth.h	Every QEP application needs to include qep_port.h
Page 213, 2 nd paragraph	file REMINDER.EXE file	REMINDER.EXE file
Page 217, 1 st paragraph	Windows GUI applications can call the PostMessage() Win32 API to queue messages and the WM_TIMER message to receive timer updates.	Windows GUI applications can call the PostMessage() Win32 API to queue messages and provide a WM_TIMER case in the window procedure to receive timer updates.
Page 222, 1 st paragraph	file DEFER.EXE file	DEFER.EXE file
Page 229, Listing 5.4, explanation section (4)	Listing 4.1	Listing 4.2
Page 234, Figure 5.11 2 nd note from bottom	... from the Alarm component	... from the AlarmClock component
Page 236, Listing 5.6	<pre>/* the HSM version of the Alarm component */</pre>	<pre>/* the FSM version of the Alarm component */</pre>
Page 239, Listing 5.8 caption	(file clock.c)	(file comp.c)
Page 244, footnote 8	must explicitly instantiate all components explicitly	must explicitly instantiate all components
Page 245, last paragraph	doorClosed_history (abbreviated to history in Figure 5.12).	doorClosed_history.
Page 247, Listing 5.9 Comment line	HSM definitio-----	HSM definition-----
Page 250, last paragraph	requires setting doorClosed_history to &ToasterOven_toasting in the exit action from "toasting" to &ToasterOven_baking in the exit action from "baking," and so on	requires setting doorClosed_history to &ToasterOven_toasting in the exit action from "toasting," and likewise to &ToasterOven_baking in the exit action from "baking," and so on
Page 261, 3 rd paragraph	easer	easier
Page 292, Figure 6.15, in "QTimeEvt"	<pre>postIn(); postEvery(); disarm();</pre>	<pre>postIn(); postEvery(); disarm(); rearm();</pre>

Page 295, 2 nd paragraph from the bottom	... code to "wonder around," silently taking care of code to "wander around," silently taking care of ...
Page 297, 1 st paragraph	(see Section 6.1.6)	(see Section 6.7.6)
Page 288, Figure 6.14	retrun-from-dispatch(e)	return-from-dispatch(e)
Page 298, Listing 6.1 explanation section (1)	When disabled, all assertion macros expand to empty statements that don't generate any code.	When disabled, all assertion macros, except <code>Q_ALLEGE()</code> , expand to empty statements that don't generate any code.
Page 300, Listing 6.1 explanation section (11)	(see [Murphy 01])	(see [Murphy 01a])
Page 309, 1 st paragraph	www.quantum-leaps.com/doc/AN_QL_Coding_Standard.pdf	www.quantum-leaps.com/resources/AN_QL_Coding_Standard.pdf
Page 310, 1 st paragraph	QP-nano in Chapter 11	QP-nano in Chapter 12
Page 312, Section 7.1.6	Perhaps that most ...	Perhaps the most ...
Page 314, 1 st paragraph	power-savings features	power-saving features
Page 315, 1 st paragraph	As all real-time frameworks, QF provides the central base class QActive...	QF provides the central base class QActive...
Page 315, left side, near bottom	Star Wars application	Fly 'n' Shoot application
Page 317, Listing 7.1, in description of "qte_darm.c"	<code>QTimeEvt_darm()</code>	<code>QTimeEvt_disarm()</code>
Page 321, Listing 7.4(1)	<code>QF_INT_LOCK_KEY</code>	<code>QF_INT_KEY_TYPE</code>
Page 326, Listing 7.7, explanation section (8)	<code>QQueue</code>	<code>QEQueue</code>
Page 326, last paragraph	See Chapter 8, "POSIX QF Port," ...	See Section 8.4, "QF Port to Linux (Conventional POSIX-Compliant OS)," ...
Page 332, Listing 7.9 explanation section (4)	The argument 'stkSto' is a pointer to the storage for the private stack, and the argument 'stkSize' is the size of that stack (in bytes), respectively.	The argument 'stkSto' is a pointer to the storage for the private stack, and the argument 'stkSize' is the size of that stack (in bytes).
Page 335, Listing 7.11 line before (6)	perform	perform
Page 336, Listing 7.11 explanation section (4)	see Chapter 6, "Customized Assertions in C and C++"	see Section 6.7.3, "Customizable Assertions in C and C++"

Page 338, last paragraph	evT_	evtT_
Page 341, explanation section (12)	QF_EPOOL_PU_()	QF_EPOOL_PUT_()
Page 342, Listing 7.13 explanation section (1)	QActive_defer() takes posts the	QActive_defer() posts the
Page 344, 2 nd paragraph code snippet	AO_ship	AO_Ship
Page 344, 3 rd paragraph	AO_ship	AO_Ship
Page 345, Listing 7.14 caption	\qpc\init\qf.h	\qpc\include\qf.h
Page 346, 1 st paragraph	QF_subsrcrList_	QF_subscrList_
Page 346, 1 st paragraph	QF_maxSignal	QF_maxSignal_
Page 348, Listing 7.17 caption	qa_pspub.c	qf_pspub.c
Page 355, Listing 7.19 explanation section (8-13)	removing a link	removing a node
Page 356, Listing 7.19 explanation section (21)	The link is advanced	The time event node pointer is advanced
Page 358, Listing 7.21 explanation section (3-8)	removing a link from	removing a node from
Page 360, 3 rd paragraph	Figure 7.8	Figure 7.9
Page 360, Figure 7.9	Counterclockwise movement...	Reverse the direction of the arrow and the text in note to "Clockwise movement..."
Page 360, Last paragraph	frequently bypass, the buffering	frequently bypass the buffering
Page 363, 2 nd paragraph	included directly in the level QActive structure	included directly in the higher-level QActive structure
Page 364, Listing 7.24 explanation section (1)	The function QActive_get_() returns a read-only (const) pointer to an event	The function QActive_get_() returns a pointer to a read-only (const) event
Page 365, Listing 7.24 explanation section (12,13)	(12,13) Additionally, a platform-specific macro...	(12) Additionally, a platform-specific macro... (13) The event pointer is returned to the caller. This pointer can never be NULL.
Page 369, Listing 7.26 explanation section (8)	you call QEQueue_get_() to post an event	you call QEQueue_postFIFO() or QEQueue_postLIFO() to post an

		event
Page 375, last paragraph	see Listing 7.12(3)	see Listing 7.12(5)
Page 377, 1 st paragraph	QPset64	QPSet64
Page 380, Figure 7.12, three occurrences	prio__	prio
Page 383, last paragraph	Listing 7.32(20)	Listing 7.32(8)
Page 384, 3 rd paragraph	Yet other class of MCUs...	Yet another class of MCUs...
Page 386, 6 th paragraph	www.quantumleaps.com	www.quantum-leaps.com
Page 387, last paragraph	QF supports many advances features ...	QF supports many advanced features ...
Page 393, Listing 8.1(12)	QFplatform-dependent QSplatform-dependent QPplatform-dependent	QF platform-dependent QS platform-dependent QP platform-dependent
Page 398 1 st paragraph	The header file <code>qep_porth.h</code> ... However, <code>qep_porth.h</code> also...	The header file <code>qep_port.h</code> ...However, <code>qep_port.h</code> also...
Page 402, 3 rd paragraph	section 8.4, "Conventional POSIX-Compliant OS (Linux)"	section 8.4, "QF Port to Linux (Conventional POSIX-Compliant OS)"
Page 406, explanation section (24)	Section 8.3	Section 8.4
Page 410, Listing 8.4 explanation section (8)	the size of that stack (in bytes), respectively	the size of that stack (in bytes)
Page 410, last line	QS-specific	OS-specific
Page 411, 3 rd paragraph	Section 8.2	Section 8.3
Page 413 1 st paragraph in Section 8.1.9	As you design you port, you must decide...	As you design your port, you must decide...
Page 414, 1 st paragraph in Section 8.2	<code>qep_porth.h</code>	<code>qep_port.h</code>
Page 415, in paragraph before Listing 8.7	exact-with	exact-width
Page 416, 2 nd paragraph	Section 8.2	Section 8.3
Page 421, 2 nd paragraph	RTOS that it is superbly documented	RTOS that is superbly documented
Page 425, 5 th paragraph	QF_EPPOL_TYPE	QF_EPOOL_TYPE

(after (9))		
Page 425, explanation section (11)	QF_EPPOL_TYPE	QF_EPOOL_TYPE
Page 430, explanation section (30)	whereas timeout	where a timeout
Page 431, 3 rd paragraph	build you own	build your own
Page 436, Listing 8.19 for lines below (6)	due to insufficient privieges	due to insufficient privileges
Page 437, Listing 8.19(17)	stops	stops
Page 438, explanation section (2)	lock in physical memory of all the pages mapped	lock in physical memory all of the pages mapped
Page 439, top of page	(6) The "ticker" thread runs... (7) The "ticker" thread calls...	(7) The "ticker" thread runs... (8) The "ticker" thread calls...
Page 439, explanation section (14)	described in triggered	described is triggered
Page 440, paragraph following (29-33)	and the rest highest priorities	and the rest of the highest priorities
Page 446, Section 9.2 title	Philosopher	Philosophers
Page 447, 1 st paragraph	your always need	you always need
Page 447, Figure 9.1 caption	Philosopher	Philosophers
Page 470, explanation section (1-3)	oversized all stacks of to 256 of 16-bit stack entries	oversized all stacks to have 256 16-bit stack entries
Page 474, 3 rd line from the top	<code>l_delay = atol(argv[1]);</code>	<code>l_delay = atol(argv[1]);</code>
Page 475, explanation section (5)	<code>t_sav</code>	<code>l_tsav</code>
Page 477, Section 9.4.1 title	In Sizing Event Queues	Sizing Event Queues
Page 477, 4 th paragraph	Listing 7.24(12-14)	Listing 7.25(12-14)
Page 484, 1 st paragraph	routed	rooted
Page 490, explanation section (10)	which as been	which has been

Page 498, Listing 10.1	<code>+-80x88\</code>	<code>+-80x86\</code>
Page 498, Figure 10.5 three occurrences	<code>prio__</code>	<code>prio</code>
Page 502, explanation section (27)	at step 13	at step 14
Page 510, explanation section (25)	could have change	could have changed
Page 510, explanation section (29)	back to step (13)	back to step (15)
Page 512, explanation section (1)	includes to the wider	includes the wider
Page 520, last paragraph	<code>QK_scheduler_()</code>	<code>QK_schedule_()</code>
Page 521, last paragraph	<code>QK_scheduler_()</code>	<code>QK_schedule_()</code>
Page 524, 5 th paragraph	<code>qep_porth.h</code>	<code>qep_port.h</code>
Page 525, 3 rd paragraph	exact-with	exact-width
Page 525, last paragraph	<code>qk_porth.h</code>	<code>qk_port.h</code>
Page 529, explanation section (1)	unconditional interrupt saving and restoring	unconditional interrupt locking and unlocking
Page 539, last paragraph	perform a lot	performs a lot
Page 540, 1st paragraph	embedded (RTE) stems	embedded (RTE) systems
Page 545, 3rd paragraph	located in the directory	located at
Page 549, last paragraph	first eight columns	first ten columns
Page 550, 3rd paragraph	$((0000135566 - 0000070346) / 7 = 65220 \approx 0x10000)$	$(0000135566 - 0000070346 = 65220 \approx 0x10000)$
Page 551, 2nd paragraph	factor of two in data density	factor of two improvement in data density
Page 552, 3rd paragraph	High Level Data Link Control	High-level Data Link Control
Page 552, 3rd paragraph	[HDLC]	[HDLC 07]
Page 562, 3rd paragraph	without entering the QS critical	without entering the QS critical section
Page 563, 4th paragraph	<code>(bimask & bit) != 0</code>	<code>(bitmask & bit) != 0</code>
Page 563, 4th paragraph	<code>(QS_glbFilter_[5] & 0x40) !</code>	<code>((QS_glbFilter_[5] & 0x40) !</code>

	= ...)	= ...)
Page 563, last paragraph	records types	record types
Page 565, last paragraph	all local filters is set	all local filters are set
Page 566, 1st paragraph	QS_BEGIN()	QS_BEGIN_NOLOCK()
Page 566, 2nd paragraph (code snippet)	<pre>#define QS_BEGIN(rec_, obj_) \</pre>	<pre>#define QS_BEGIN_NOLOCK(rec_, obj_) \</pre>
Page 567, paragraph 3, (item 2)	Following the Fame Sequence Number	Following the Frame Sequence Number
Page 567, paragraph 5, (item 4)	over the frame Sequence Number	over the Frame Sequence Number
Page 567, paragraph 6, (item 5)	HDLC flag	HDLC Flag
Page 568, 3rd paragraph	over the Fame Sequence Number	over the Frame Sequence Number
Page 569, Section 11.3.7, 2 nd paragraph	You can employ just about any repetition physical data link available...	You can employ just about any physical data link available...
Page 569, 4th paragraph	Your can apply	You can apply
Page 570, Listing 11.7 caption	QS_initBuf(QS_initBuf()
Page 571, 1st paragraph	options to avid losing	options to avoid losing
Page 575, 4th paragraph	OS_onFlush()	QS_onFlush()
Page 580, 2nd paragraph	QK, C/OS-II, and Linux	QK, uC/OS-II, and Linux
Page 580, 3rd paragraph	functions such as QS_onInit()	functions such as QS_onStartup()
Page 580, Listing 11.11 caption	qp_port.h	qs_port.h
Page 581, explanation section (5)	qf_port.h	qs_port.h
Page 594, Table 11.4 Header (2nd row)	Timesstamp	Timestamp
Page 595, explanation section (2)	l_pholo_0_	l_philo_0_
Page 601, 5th paragraph	8284 timer/counter	8254 timer/counter
Page 601, 6th paragraph	8284 timer/counter	8254 timer/counter

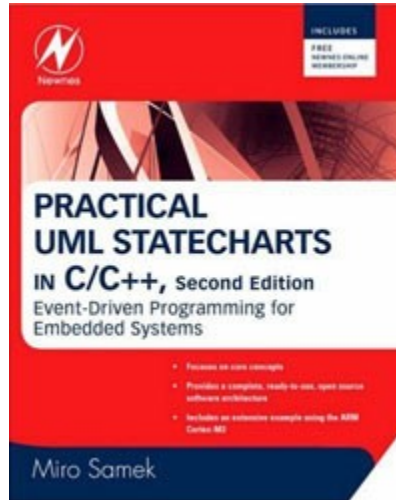
Page 605, explanation section (10)	so it does need to have	so it does not need to have
Page 607, explanation section (3)	formatted as 1 using one digit	formatted as using one digit
Page 611, 2nd paragraph	[Turely 02]	[Turley 02]
Page 617, 4th paragraph	The order or the active object control blocks	The order of the active object control blocks
Page 618, explanation section (13-15)	function must first explicitly calls	function must first explicitly call
Page 619, explanation section (8)	The <code>qpn_port.h</code> must include	The <code>qpn_port.h</code> header file must include
Page 619, explanation section (9)	The <code>qpn_port.h</code> must include	The <code>qpn_port.h</code> header file must include
Page 620, 3rd paragraph	Listing 12.2(2)	Listing 12.2(1)
Page 621, explanation section (3-5)	Listing 12.1(12)	Listing 12.1(7)
Page 626, explanation section (15)	overflow the dynamic range	overflow the range
Page 626 & 627: Listing 12.5, three occurrences	<code>return (QState)0;</code>	<code>return Q_HANDLED();</code>
Page 627: Listing 12.5, next to last line	<code>return (QState)&Tunnel_active;</code>	<code>return Q_SUPER(&Tunnel_active);</code>
Page 630, last paragraph	<code>C:\software\qpn</code>	<code><qpn>\qpn</code>
Page 631, 1st paragraph	derivation of concrete active objects	derivation of concrete active object classes
Page 632, Figure 12.3, in the "QHsm" class	<code>QState Handler</code>	<code>QStateHandler</code>
Page 632, last paragraph	Every QP-application	Every QP-nano application
Page 633, Listing 12.7	Platform-specific QP examples	Platform-specific QP-nano examples
Page 633, Listing 12.7	<code>+main.c -</code>	<code>+main.c - main() entry point</code>
Page 634, Listing 12.7	<code>- QP-nano Reference Manual"</code>	<code>- "QP-nano Reference Manual"</code>
Page 640, 3rd paragraph	deriving application-specific active objects	deriving application-specific active object classes

Page 648, 6th paragraph (just above (9))	Such as global variable	Such a global variable
Page 650, explanation section (3)	The advance policy	The advanced policy
Page 652, explanation section (2)	log2(bmask)	log2(bitmask)
Page 652, explanation section (3)	QF_readSet_	QF_readySet_
Page 657, explanation section (6)	The QK_SCHEDULE_() encapsulates	The QK_SCHEDULE_() macro encapsulates
Page 659, explanation section (6)	All active objects in the application are initialized, exactly the same way as in 12.16(6--11).	All active objects in the application are initialized, the same way as in 12.16(6--11).
Page 661, Listing 12.19, the comment between (12) and (13)	set cb and a again	set cb and act again
Page 669, explanation section (3)	sperstate	superstate
Page 671, 5th paragraph	PED_WAITING	PEDS_WAITING
Page 671, last paragraph	PED_WAITING	PEDS_WAITING
Page 675, 2nd paragraph	When you apply low-power mode is MSP430	When you apply low-power mode in the MSP430
Page 686, 2nd paragraph	Figure B.1C	Figure B.1(C)
Page 693, 5th entry	[Butenhof 97] ... [Butenhof 97] ...	[Butenhof 97] ... (unintended repetition)
Page 695, two occurrences on the same line	Kerninghan	Kernighan
Page 695		[Meyer 97b] Bertrand Meyer. Letters from readers (response to the article "Put it in the contract: The lessons of Ariane" by Jena-Marc J'ez'l'equel, and Bertrand Meyer). IEEE Computer, 30(2):8--9, 11, 1997.
Page 696	Rambaugh, James	Rumbaugh, James

1.3 Contact Information

Quantum Leaps, LLC
103 Cobble Ridge Drive
Chapel Hill, NC 27516
USA

+1 866 450 LEAP (toll free, USA only)
+1 919 869-2998 (FAX)
e-mail: info@quantum-leaps.com
WEB : <http://www.quantum-leaps.com>
<http://www.state-machine.com>



“Practical UML Statecharts in C/C++, Second Edition: Event Driven Programming for Embedded Systems”, by Miro Samek, Newnes, 2008

