AN413
Application note

Initialization of the ST9

1 Introduction

The ST9 family offers the microprocessor designer a wide variety of architectural features configurable to the user's specific application requirements. Central to all these configurations is a multiple register based microcomputer core to which may be added on-chip, powerful peripheral components including A/D converters, Serial Communication Interface units (SCIs), and 16-bit Multifunction timers with input capture/output compare capabilities. The availability, on-chip, of these application-specific units obviates the need for external interface design as well as offering high-speed and good reliability.

The particular peripherals incorporated on-chip may themselves be individually configured to offer a wide variety of functional (architectural) alternatives. This configuration is typically implemented by simple software routines included in the power-on- or system- reset routines. The sole difficulty which the user may initially encounter stems, in fact, from the power and versatility of this approach to system design. The large number of available options means that the user must specify a large number of system parameters by initializing control register contents for the specific peripheral units.

The objective of this Application Note is to suggest to the user a programming structure and philosophy to aid in the initial configuration of the system. The approach is illustrated by a number of specific examples selected from the wide range available for the ST9030, ST9040 families, but are applicable to all ST9s.

Contents

List of tables

2 System reset

After processor Reset the control and status registers, located on the group F pages (0-63) are forced to preset values which define a default Reset configuration for the ST9 system. By way of example the internal clock frequency (INTCLK) is set to the internal crystal oscillator (or externally applied clock frequency, if supplied) divided by two without prescaling, and the individual pins of Parallel Ports 0,1, and 6 are set to bidirectional Pull-up mode (for systems with on-chip ROM). On releasing the external RESET signal the processor PC is loaded with the contents of the Reset Vector stored in address locations 0 and 1. This causes a jump to a Reset routine in which the designer may reconfigure the ST9 system as appropriate to the requirements of his particular application, by loading suitable values into the system registers.

The number of registers to be initialized may be considerable for a representative ST9 system. Additionally, the application-specific interrupt routines will, in general, involve the manipulation of substantial system resources, e.g. read/write of data registers, and test/reset of status, mask, and control registers. The associated programming task may appear daunting in prospect on first acquaintance with the ST9 system.

Conceptually, the organization of the associated software is relatively simple and straightforward as may be recognized by grouping under four headings the programming steps involved in the initialization of ST9 peripherals and the organization of interrupt service routines.

a) ST9 core system configuration

Certain core system resources are common to all on-chip peripherals and may be specified in a common routine which is invoked at System Reset. Such common resources include clock configuration, system and user stack specification, global interrupt masking, processor priority setting, parallel port bit-by-bit specification, and setting of external memory wait-cycles. The setting up of the interrupt vector table, and certain global masking or enabling operations, may also be included under this heading.

b) Individual on-chip peripheral configuration

The configuration of on-chip peripherals, e.g. Multifunction Timers, A/D Converters, etc., involves the loading of suitable bit-patterns into group F page registers. This enables the specification of input and output signals, determination of the peripheral's mode of operation, and the selection of internal or external clock and control signals.

c) Individual on-chip peripheral initialization

The initialization of a particular on-chip peripheral may involve the setting or clearing of device-specific enable and masking bits, specification of interrupt priority levels, clearing of status/flag values, and the loading of data and/or limit registers.

d) Organization of interrupt service routines

This will normally include context-saving and restoring of the PC and system status, plus the working-register and page-pointer registers, together with the values of any working registers used in the routine. The routine proper may include testing of status flag bits, and the reading and writing of data registers associated with the particular

device. Finally, the interrupt pending bits should be cleared, the context restored, and individual masking and enabling bits restored to the appropriate values.

In practical programming terms there will normally be a single routine invoked on system RESET which carries out the core system configurations listed under heading a) above. For each individual peripheral there will typically be a single routine which carries out the configuration and initialization operations listed under headings b) and c). There will also be one or more interrupt routines associated with each peripheral, e.g. the A/D converter may require in general two interrupt routines, one for End of Conversion, and one for out of range operation (i.e. Analog Watchdog operation) on channels 6 and 7.

An example of a core-system configuration is given in Appendix B, and Appendices C,D,E, and F give configuration/initialization examples, and Interrupt routines for the Timer, A/D Convertor, SCI unit, and Timer/Watchdog respectively.

There is not enough space in a short note to discuss these programmes in detail on a line by line basis. Instead the approach will be to list, for each device, the resources which need to be taken into consideration when configuring, initializing, and servicing the particular device. An example will then be given of the specific use of each such resource. With this background, the interested user should be able to follow in detail those listings most relevant to his particular application area.

3 ST9 basic system configuration

Table 3 and *Table 3* in *Appendix A* list the registers which should be loaded with specified bit-patterns in order to initialize the ST9 to a basic system configuration. A demonstration routine which carries this out for a representative ST9 system is listed in *Appendix B*. The main routine, RESET_START, is invoked at system Reset. Also shown in Appendix B are the Assembler Declarations and directives which enable the Interrupt Vector Address Table to be set up in program memory.

3.1 The vector address table

The ST9 implements an interrupt vectoring structure that allows the on-chip peripheral to identify the location of the first instruction of the Interrupt Service Routine (ISR). Each interrupt module has a specific Interrupt Vector Register (IVR) mapped on the register file pages. When the interrupt request is acknowledged, the peripheral interrupt module provides, via the IVR, the vector to point to the address of the Interrupt Service Routine in the Vector Table.

The Interrupt Vector table containing the list of addresses of the Interrupt Service Routine must be located in the first 256 locations of program memory. The first 6 locations of Program memory are reserved as follows:

Table 1. Reserved locations of program memory

Address	Content
0	Address high of Power on Reset routine
1	Address low of Power on Reset routine
2	Address high of Divide by Zero Trap Subroutine
3	Address low of Divide by Zero Trap Subroutine
4	Address high of Top Level ISR
5	Address low of Top Level ISR

Note that since the above locations are fixed by the hardware no associated IVR register is involved. For certain interrupt modules more than one interrupt routine may be required. For example the A/D Converter has separate interrupts for the End of Conversion and Channel 6/7 analog underflow/overflow conditions.

In such cases the IVR register specifies the more significant, and the interrupt module hardware specifies the least significant bits of the Vector Table address.

The following Assembler outline shows how the corresponding Vector table entries may be established.

```
ADC_IT_VECT:= 30h
.
.org ADC_IT_VECT
.word ADC_WDG
```

```
   .word ADC_EOC
.
ADC_WDG:
   ; Code for the Analog Watchdog Routine is included here
   ; Note that in the example in Appendix B
   ; the System Reset routine is invoked for out of
   ; range conditions on Channels 6 and 7
   .
   iret
ADC_EOC:
   ; End of A/D conversion interrupt routine included here
   iret
```

3.2 Port initialization

The ST9 has up to a maximum of 64 lines dedicated to input/output. These lines, grouped into eight 8-bit ports, can be independently programmed to provide parallel input/outputs with or without handshake or may be used to connect in/out signals to/from the peripherals (e.g. Core, Timers, SCI units, etc.) present on the chip. The functional allocation of the Ports to support system tasks may be summarized as follows:

Table 2. Port functional allocations

Port	Functions
0	Usable as I/O Port (without handshake) or as multiplexed low-address and data lines for external memory.
1	Usable as I/O Port (without handshake) or as high-address lines for external memory.
2	Usable as I/O Port (without handshake) or for SPI functions; Also INT1, INT2, and INT3 inputs.
3	Usable as I/O Port (without handshake) or for Timer functions.
4	Usable as I/O Port (with or without handshake)
5	Usable as I/O Port (with or without handshake).
6	Usable as I/O Port (without handshake)
7	Usable as I/O Port (without handshake) or for SCI functions. Also used for INT4, INT5, and INT6 inputs or for Control signals for slow external memory

Ports 0, 1, and 6 are automatically initialized on system Reset to correspond to the installed on-chip memory.

Ports 2, 3, 4, 5, 6, and 7 need to be initialized (if available) to satisfy the specific application requirements

for external I/O, plus any alternative function assignments of port pins, and internal interconnections. *Table 5*,*Appendix A*, lists the complete set of Port Configuration registers together with their addresses.

Example:

```
C7 0A       spp     P3C_PG
F5 FC 05    ld      P3C0R,#00000101b
F5 FD 0F    ld      P3C1R,#00001111b
F5 FE 05    ld      P3C2R,#00000101b
```

In this example Port 3 pins 4, 5, 6, and 7 are configured as bidirectional pins, with weak pull-up output and TTL inputs. Pins 0 (T0INA) and 2 (T0INB) are configured as TTL inputs, and Pins 1 (T0OUTA) and 3 (T0OUTB) are configured as Alternate Function Push-pull outputs.

3.3 Multifunction timer configuration

The ST9 Multifunction Timer is configured by loading suitable control-bit patterns in the group F page register TCR, TMR, ICR, OACR, and OBCR (see *Table 6* in *Appendix A*). Note that registers EIMR and CICR provide global control functions common to all on-chip peripherals and are hence initialized conveniently in the basic system configuration routine.

The External Input Control Register, ICR, controls input source selection (internal/external), input mode selection (falling/rising edge sensitive, etc.), counter mode of operation (continuous, one-shot, etc.), and input function (Gate, Trigger, up/down control, etc.).

Example:

```
F5 FA 54 ld T_ICR,#01010100b
```

This instruction selects the external input A as a falling-edge-sensitive Trigger input, and the B input is a normal Port I/O pin.

The Multifunction Timer Control Register, TCR, controls counter clear and prescaler reload operations as well as providing a counter enable control bit and counter status flags.

Example:

```
F5 F8 48 ld T_TCR,#01001000b
```

This instruction halts the counter operation but provides for subsequent UP counting with counter clear and Prescaler reload on Reg0 or Reg1 capture.

The Multifunction Timer Mode Register, TMR, selects the clock source for the counter-prescaler input, enables Retrigger or Continuous mode, and controls register load/capture operations.

Example:

```
98 8C ld T_TMR,#10001100b
```

This pattern enables output 1 and disables output 0, disables bi-value modes, and selects Reg0 for capture and Reg1 for monitor. Retriggerable continuous mode is selected.

The Output Control Register, OACR, links the output T0OUTA to counter overflow/underflow and Compare events, and provides for subsequent Set, Reset, or Toggle of the external output. The on-chip event (OCE) may be linked to a COMP0 event.

Example:

```
F5 F5 1B ld T_OACR,#00011011b
```

In this example T0OUTA is preset to 1, and is subsequently set by COMP0, toggled by COMP1, and Reset by OVF. The OCE signal is generated by a successful CMP0 compare event.

The Output Control Register, OBCR, links the output T0OUTB to counter overflow/underflow and Compare events, and provides for subsequent Set, Reset, or Toggle of the external output. The on-chip event (OCE) may be linked to a counter overflow/underflow event.

Example:

```
F5 F6 83 ld T_OBCR,#10000011b
```

In this example T0OUTB is preset to 1, and is subsequently reset by COMP0, and set by OVF and COMP1.

The OCE signal is generated by a counter overflow/underflow event.

3.4 Multifunction timer initialization

Initialization of the Multifunction Timer requires loading of the Prescaler register and the two Comparison registers. The timer Status register should be cleared, the Vector Table entry should be set, and the Multifunction Timer counter actions enabled. The interrupt/DMA priority levels should be set and the mask bits should be adjusted as appropriate to the application. Further, if DMA operations are specified, DMA address and counter registers will require initialization.

The Prescaler Register, PRSR, holds the preset value for the 8-bit prescaler.

Example:

```
BC 00 ld T_PRSR,#00h
```

This defines a division ratio of 1 and the maximum counter clock is generated (INTCLK/3).

The Multifunction Timer Flags Register, FLAGR, contains flags which register successful capture or comparison events together with OVF/UNF and overrun conditions.

Example:

```
15 FE FD and T_FLAGR,#~ocm0
```

This example resets the overrun bit for COMP0 operations.

The Interrupt Vector Register, IVR, should be loaded with the 5 most significant bits of the Multifunction Timer's interrupt vector address in program memory. The interrupt source (compare, capture, or OVF/UNF) provides the least significant 3 bits to provide the correct vector link.

Example:

```
F5 F2 10 ld T0_IVR,#T0_IT_VECT
```

In this example IVR is loaded with the start address (10h) of the block of 8 words in the vector table allocated to the 5 different Multifunction Timer interrupts.

The Interrupt/DMA Control Register, IDCR, is used to set the Interrupt and DMA priority levels, and the DMA transfer source and destination. It also enables Swap mode and contains End of Block condition flags.

Example:

```
F5 F3 D6 ld T0_IDCR,#11000110b
```

In this example the priority level is set at a value of 6, and the Swap mode is disabled. The DMA capture channel source is REG0, and the DMA compare channel source is CMP0.

The Interrupt/DMA Mask Register, IDMR, contains a global Multifunction Timer Interrupt enable plus individual DMA and Interrupt enable bits for overflow as well as successful capture and comparison events.

Example:

```
F5 FF 04 ld T_IDMR,#00000100b
0F FF 80 or T_IDMR,#gtien
```

The first instruction sets the interrupt enable on CMP0, and the second instruction globally enables all Multifunction Timer interrupts.

The DMA Counter Pointer Register, DCPR, defines the DMA area and source, and specifies the location of the DMA length register.

Example:

```
F5 F0 4C ld T0_DCPR,#CPT_LG_DMA
```

The DMA length register is 4Ch = rr12 = RR76 and the transfer occurs to/from Program/Data memory.

The DMA Address Pointer Register, DAPR, defines the DMA area and source, and specifies the location of the DMA address register.

Example:

```
F5 F1 48 ld T0_DAPR,#CPT_AD_DMA
```

The DMA address register is 48h = rr8 = RR72. In conjunction with the DPCR value in the above example it specifies Program memory for the buffer.

3.5 A/D converter configuration/initialization

Configuration of the A/D converter requires loading of 4 registers only, CLR, CRR, ICR, and IVR (*Table 9*), and initialization of this device involves, apart from global masking, loading of two double (threshold registers). Hence a single routine can be written to cover both the configuration and initialization aspects of A/D Converter use.

The Control Logic Register, CLR, defines the Analog channel conversion start address, selects internal/ external triggers, and enables continuous or single conversion and power up/down modes. This register also contains a start/stop status/control bit.

Example:

```
F5 FD 04 ld AD_CLR,#00000100b
```

In this example, the conversion scan starts with channel 0 when enabled, powers up the A/D convertor, halts conversion, and specifies single conversion scan mode.

Please note that before enabling any A/D conversion, it is mandatory to set the low bit of Control Logic Register at least 60ms before the first conversion start. This is in order to correctly bias the analog section of the converter.

The Interrupt Vector Register, IVR, defines the most significant 6 bits of the vector table byte address. It thus points to the first of two word addresses which correspond to the analog watchdog and End of conversion interrupt routines.

Example:

```
F5 FF 32 ld AD_IVR,#ADC_ITEOC_VECT
```

In this example, an address of 50 (decimal) is loaded into IVR. Hence a subsequent A/D convertor EOC interrupt will cause a Vector Table access at location 50.

The Interrupt Control Register, ICR, contains the priority level specification, the two source interrupt flags (Analog Watchdog and EOC) and their individual masking bits.

Example:

```
F5 FE 20 ld AD_ICR,#00100000b
05 FE 20 or AD_ICR,#00000110b
```

In this example, the priority level is first set at 0, End of Conversion interrupts are enabled, and the Analog Watchdog interrupt is masked. The second instruction then sets the priority to a level of 6.

If the Analog Watchdog is enabled (bit 6 in ICR) it will be necessary to load the threshold registers for channels 6 and 7. In this case access will be made in the interrupt routine to register CRR.

The Compare Result Register, CRR, contains 4 flags showing the results of comparison operations between the current values of data registers 6 and 7, and the upper and lower threshold registers.

3.6 SCI unit configuration

The list of registers to be initialized when configuring the SCI unit is given in *Table 11*. The functions of these registers, and some illustrative examples of their use, are as follows:

The Character Configuration Register, CHCR, is used to define the serial frame format.

Example:

```
AC E3 ld S_CHCR,#E3h
```

This example defines a serial frame as follows: 8 data bits, 1 stop bit, even parity, and address input if the character matches the contents of the Address Register.

The Clock Configuration Register, CLCR, is used to specify the transmitter, receiver, and Baud Rate clock sources, and the clock divisor ratio. It also enables Auto Echo and Loopback test modes.

Example:

```
BC 80 ld s_clcr,#txclk
```

In this example, the Transmitter and Receiver clocks are provided by the Baud Rate Generator. Each data bit period will be 16 clock periods (asynchronous mode), and the Auto Loop and Loopback modes are disabled.

The Baud Rate Generator Register, BRGR, specifies a 16-bit division ratio.

Example:

```
BF DC 00 4E ldw s_brgr,#DIV_9600
```

This example specifies a division ratio yielding 9600 Bauds with a 24 Mhz external clock.

Writing to a Baud Rate Generator Register immediately disables and resets both the SCI Baud Rate generator, the transmitter and receiver circuitry. After writing to the remaining Baud Rate Generator Register, the transmitter and receiver circuits are enabled. The Baud Rate Generator will load the new value and start counting.

To initialize the SCI, user should first initialize one Baud Rate Generator Divisor Register. This will reset all SCI circuitry. Initialize all other SCI registers for the desired operating mode. To enable the SCI, initialize the remaining Baud Rate Generator Register.

The Address Compare Register, ACR, contains an 8-bit value which may be used as a match against which a received address may be tested to set the Receive Address Pending bit.

Example:

```
5C 0D ld s_acr,#RETURN
```

This will cause the Receive Address Pending bit to be set if an End of Command character bit-pattern is received.

The Interrupt Vector Register, IVR, defines the most significant 5 bits of the vector table byte address. It thus points to the first of four vector table word address entries.

Example:

```
4C 00x ld s_ivr,#SCI_IT
```

In this example, after the external symbol has been linked in, the Vector Table entry address will be loaded into IVR at execution time.

The Interrupt Mask Register, IMR, contains five interrupt masking bits and two End of Block DMA status bits. It also selects the shift register or holding register as source of the transmitter register empty interrupt.

Example:

```
6C 05 ld s_imr,#00000101b
```

In this example the interrupt pending bits are reset, the Transmitter data interrupt is masked, and the Receiver data, data error, and address interrupts are unmasked.

The Interrupt/DMA Priority Register, IDPR, specifies the Interrupt/DMA priority, selects one of four Address modes, and controls the emission of Break characters and enables address/9th bit data mode. It also provides mask bits for Receive and Transmit DMA transfers.

Example:

```
9C 04 ld s_idpr,#04h
```

In this example a priority level of 4 is specified, and Transmitter DMA requests are masked.

3.7 SCI unit initialization

The list of registers to be initialized when initializing the SCI unit is given in *Table 12*. The functions of these registers, and some illustrative examples of their use, are as follows:

The Receiver DMA Transaction Counter Pointer Register, RDCPR, contains the register file address of the receiver DMA transaction counter. In addition it determines whether the DMA transfers occur in the register file or in memory.

Example:

An example of the use of this register is provided below (see RDAPR example).

The Receiver DMA Destination Address Pointer Register, RDAPR, contains the register file address of the receiver DMA data destination. In addition, in conjunction with bit 0 of RDCPR, it determines whether the DMA transfers occur in Program or Data memory.

Example:

```
00 FF LNG-DMA_SCI := 0Fh

00 A0 DEPART_DMA_SCI := 0A0h

00 02 NUM_RDAP := 2

00 03 NUM_RDCP := 3

2C 03 ld S_rdcpr,#NUM_RDCP

1C 02 ld S_rdapr,#NUM_RDAP

F5 03 0F ld R#NUM_RDCP,#(LNG_DMA_SCI)

F5 02 00 ld R#NUM_RDAP,#(DEPART_DMA_SCI)
```

In this program sequence the DMA transaction counter and Address Pointer register addresses are defined to be R3 and R2 respectively. These two registers are initialized for a block of size 15 bytes starting at register address A0, i.e. R160.

The Transmitter DMA Transaction Counter Pointer Register, TDCPR, contains the register file address of the transmitter DMA transaction counter. In addition it determines whether the DMA transfers occur in the register file or in memory.

Example:

An example of the use of this register is provided below (see TDAPR example).

The Transmitter DMA Destination Address Pointer Register, TDAPR, contains the register file address of the transmitter DMA data destination. In addition, in conjunction with bit 0 of TDCPR, it determines whether the DMA transfers occur in Program or Data memory.

Example:

```
00 FF LNG-DMA_SCI := 0Fh

00 A0 DEPART_DMA_SCI := 0A0h

00 06 NUM_TDAP := 6

00 07 NUM_TDCP := 7

2C 07 ld S_TDCPR,#NUM_TDCP

3C 06 ld S_TDAPR,#NUM_TDAP
```

```
F5 07 0F ld R#NUM_TDCP,#(LNG_DMA_SCI)
F5 06 00 ld R#NUM_TDAP,#(DEPART_DMA_SCI)
```

In this program sequence the DMA transaction counter and Address Pointer register addresses are defined to be R7 and R6 respectively. These two registers are initialized for a block of size 15 bytes starting at register address A0, i.e. R160.

3.8 Timer/watchdog unit configuration

Configuration of the Timer/Watchdog requires loading of the 6 registers listed in *Table 13*, *Appendix A*.

The Timer/Watchdog Control Register, WDTCR, contains a start/stop bit, and is also used to select input, output, and counter modes, as well as input and output enable bits.

Example:

```
BC 80 ld wdtcr,#80h
```

In this example the Timer starts counting down in continuous mode, and the input and output sections are disabled.

The Wait Control Register, WCR, as well as specifying the number of wait states for access to off-chip program and data memory enables the Watchdog function.

Example:

```
CC 40 ld wcr,#wden
```

In this example the Watchdog action is disabled, and the number of wait states are set to zero.

The External Interrupt Vector Register, EIVR, contains a bit, TLIS, which is used to control the Top Level Interrupt source (Timer/Watchdog EOC or External NMI). A second bit IAOS is used to select the Timer/Watchdog as an interrupt source on channel A0 (INT0). This register is also used to supply the 4 most significant bits of the External Interrupt Vector.

Example:

```
6C 20 ld eivr,#EXT_IT_VECT
```

In this example the Timer/Watchdog EOC generates an interrupt on channel A0 at each End of Count. The Top Level Interrupt is isolated from the NMI input and may be used for a Software Trap.

The Timer/Watchdog Prescaler Register, WDTPR, contains an 8-bit value which is loaded into the Prescaler register.

Example:

```
90 DA clr wdtpr
```

The specified Prescaler value of zero leads to a minimum timer count period of 333ns, assuming a system clock running at 12MHz.

The Timer/Watchdog High Register, WDTHR, and Timer/Watchdog Low Register, WDTLR, together contain a 16-bit value which is loaded into the counter at each End of Count.

Example:

```
BF F8 0B BB ldw WDTR,#3003
```

The specified count value leads to a count period of about 1 millisecond, (3003 x 333ns).

3.9 Timer/watchdog unit initialization

The External Interrupt Priority Level Register, EIPLR, specifies the priority level of four pairs of external interrupts, a), A1,...D0, D1. It is thus used to set the priority of the Timer/Watchdog EOC interrupt routine, called via channel A0.

Example:

```
5C FE ld eiplr,#0FEh
```

In this example priority levels of 4 and 5 are specified for the pair INTA0, INTA1.

The External Interrupts Pending Bit Register, EIPR, holds the eight interrupt pending bits for the external interrupts, including, in the present context, the Watchdog/Timer EOC interrupt. These bits are set by hardware action and reset by software during the service routine.

Example:

```
90 D3 clr eipr
```

In this example all the external interrupt pending bits are cleared.

The External Interrupts Mask-Bit Register, EIMR, holds the eight interrupt mask bits for the external interrupts, including, in the present context, the Timer/Watchdog EOC interrupt.

Example:

```
4C 01 ld eimr,#ia0
```

In this example the Timer/Watchdog End of Count on Channel A0 is unmasked.

3.10 Interrupt service routine organization

When an enabled interrupt is acknowledged the Interrupt machine cycle performs the following actions:

1. All maskable interrupts are disabled by clearing the EI bit of register CICR.
2. The PC (two bytes) and the FLAGS register are saved on the System stack.
3. The PC is loaded with the 16-bit vector stored in the Vector Table.

On exit from the Interrupt service, using an IRET instruction the following operations are carried out:

4. The FLAGR register is restored from the System stack.
5. The PC is restored from the System stack.
6. The unmasked interrupts are enabled by setting the CICR.EI bit.

In general additional resources must be saved and restored apart from those handled automatically by the system as listed above. In a typical case these additional resources will include the two Register pointer registers, the Page-pointer register, and any working registers used in the Interrupt routine.

An outline for a suitable Interrupt service routine is hence as follows:

```
Label_int:
   work_reg_page0 = (0Dh*2)
```

```
work_reg_page1 = (0Dh*2) + 1
WDT_PG = 0
T0c_PG = 9
T0d_PG = 10
S_PG = 24
AD0_PG = 63
push RP0
push RP1
push PPR
spp #T0d_PG
srp0 #work_reg_page0
srp1 #work_reg_page1
push r0
push r1
push rA
   ;
   ;
   ;Interrupt Service routine
   ;appears here, including
   ;read/write data registers
   ;test status flags
   ;clear interrupt pending flags
pop rA
pop r1
pop r0
pop PPR
pop RP1
pop RP0
iret
```

4 Summary

This application note has attempted to formalize and simplify the programming task of configuring and initializing an ST9 system. The resources to be controlled have been listed with brief examples of their use.

Complete examples of ST9 configuration, initialization, and Interrupt Service routines are presented in a set of Appendices. These programs have been written for an ST9030 but can be readily adapted where necessary for use with other versions.

5 References

(1) “ST9 Technical Manual”, STMicroelectronics.

(2) Application Note AN411, SYMBOLS.INC Standard Definitions of ST9 Registers and Register-bits.

Appendix A ST9 core and peripheral configuration/initialization

Table 3. System configuration: system registers

Mnemonic	Name	Register	Hex.	Page	Reset value (hex)
CICR	Central Interrupt Control Register	R230	E6	-	87
FLAGR	Flags Register	R231	E7	-	XX
RP0R	Register Pointer 0	R232	E8	-	XX
RP1R	Register Pointer 1	R233	E9	-	XX
PPR	Page Pointer Register	R234	EA	-	XX
MODER	Mode Register	R235	EB	-	E0
USPHR	User Stack Pointer (high)	R236	EC	-	XX
USPLR	User Stack Pointer (low)	R237	ED	-	XX
SSPHR	System Stack Pointer (high)	R238	EE	-	XX
SSPLR	System Stack Pointer (low)	R239	EF	-	XX

Table 4. System configuration: page registers

Mnemonic	Name	Register	Hex.	Page	Reset value (hex)
EECR	EEPROM Control Register Mask Register	R241	F1	0	87
EITR	External Interrupt Trigger-Event Register	R242	F2	0	XX
EIPR	External Interrupt Pending Register	R243	F3	0	XX
EIMR	External Interrupt Mask Register	R244	F4	0	XX
EIPLR	External Interrupt Priority Level Register	R245	F5	0	XX
EIVR	External Interrupt Vector Register	R246	F6	0	E0
NICR	Nested Interrupt Control Register	R247	F7	0	XX
WCR	Wait Control Register	R252	FC	0	7F

Table 5. Port configuration registers

Port	Name	Register	Hex	Page (hex)
0	Data Register Control Registers (PxC0-PxC2)	R224 R240-R242	E0 F0-F2	- 2
1	Data Register Control Registers (PxC0-PxC2)	R225 R244-R246	E1 F4-F6	- 2
2	Data Register Control Registers (PxC0-PxC2) Handshake Control Register	R226 R248-R250 R251	E2 F8-FA FB	- 2 2
3	Data Register Control Registers (PxC0-PxC2) Handshake Control Register	R227 R252-R254 R255	E3 FC-FE FF	- 2 2
4	Data Register Control Registers (PxC0-PxC2) Handshake Control Register	R228 R240-R242 R243	E4 F0-F2 F3	- 3 3
5	Data Register Control Registers (PxC0-PxC2) Handshake Control Register	R229 R244-R246 R247	E5 F4-F6 F7	- 3 3
6	Data Register Control Registers (PxC0-PxC2)	R251 R248-R250	FB F8-FA	3 3
7	Data Register Control Registers (PxC0-PxC2)	R255 R252-R254	FF FC-FE	3 3

Reset values:

Ports 2, 3, 4, and 5: PcX0: 00000000

PcX1: 00000000

PcX2: 00000000

Handshake Control Registers: 111111111

Table 6. Multi-function timer configuration/initialization registers (MFT0)

Mnemonic	Name	Registers	Hex	Page (Hex)	Reset value (binary)
CICR	Central Interrupt Control Register	R230	E6	-	10000111
TCR	Timer Control Register	R248	F8	10	00000XXX
TMR	Timer Mode Register	R249	F9	10	0
ICR	External Interrupt Control Register	R250	FA	10	0000XXXX
OACR	Output A Control Register 0	R252	FC	10	XXXXXX0X
OBCR	Output A Control Register 1	R253	FD	10	XXXXXX0X
IDMR	Interrupt/DMA Mask Register	R255	FF	10	0
DCPR	DMA Counter Pointer Register	R240	F0	9	XXXXXXXX
DAPR	DMA Address Pointer Register	R241	F1	9	XXXXXXXX
IVR	Interrupt Vector Register	R242	F2	9	XXXXXXXX
IDCR	Interrupt/DMA Control Register	R243	F3	9	11000111

Table 7. Timer data/status registers (MFT0)

Mnemonic	Name	Registers	Hex	Page (Hex)	Reset value (binary)
REG0HR	Capture/Reload Register 0 (High)	R240	F0	10	XXXXXXXX
REG0LR	Capture/Reload Register 0 (Low)	R241	F1	10	XXXXXXXX
REG1HR	Capture/Reload Register 1 (High)	R242	F2	10	XXXXXXXX
REG1LR	Capture/Reload Register 1 (Low)	R243	F3	10	XXXXXXXX
CMP0HR	Compare Register Register 0 (High)	R244	F4	10	XXXXXXXX
CMP0LR	Compare Register Register 0 (Low)	R245	F5	10	XXXXXXXX
CMP1HR	Compare Register Register 1 (High)	R246	F6	10	XXXXXXXX
CMP1LR	Compare Register Register 1 (Low)	R247	F7	10	XXXXXXXX
PRSR	Prescaler Register	R251	FB	10	0
FLAGR	Timer Flags Register	R254	FE	10	0

Table 8. Configuration/initialization registers

Mnemonic	Name	Register	Hex.	Page	Reset Value (binary)
CRR	Compare Result Register	R252	FC	63	1111
CLR	Control Logic Register	R253	FD	63	0
ICR	Interrupt Control Register	R254	FE	63	1111
IVR	Interrupt Vector Register	R255	FF	63	XXXXXX10

Table 9. A/D channel registers

Mnemonic	Name	Register	Hex.	Page
AD_D0R	Channel 0 Data Register	R240	F0	63
AD_D1R	Channel 1 Data Register	R241	F1	63
AD_D2R	Channel 2 Data Register	R242	F2	63
AD_D3R	Channel 3 Data Register	R243	F3	63
AD_D4R	Channel 4 Data Register	R244	F4	63
AD_D5R	Channel 5 Data Register	R245	F5	63
AD_D6R	Channel 6 Data Register	R246	F6	63
AD_D7R	Channel 7 Data Register	R247	F7	63

Table 10. A/D threshold registers

Mnemonic	Name	Register	Hex.	Page
AD_LT6R	Channel 6 Lower Threshold Register	R248	F8	63
AD_UT6R	Channel 6 Upper Threshold Register	R249	F9	63
AD_LT7R	Channel 7 Lower Threshold Register	R250	FA	63
AD_UT7R	Channel 7 Upper Threshold Register	R251	FB	63

Table 11. SCI configuration registers

Mnemonic	Name	Register	Hex.	Page	Reset value (binary)
IVR	Interrupt Vector Register	R244	F4	24	XXXXXXXX
IMR	Interrupt Mask Register	R246	F6	24	0XX00000
ISR	Interrupt Status Register	R247	F7	24	XXXXXXXX
IDPR	Interrupt/DMA Priority Register	R249	F9	24	XXXXXXXX
CHCR	Character Recognition Register	R250	FA	24	XXXXXXXX
CCR	Clock Configuration Register	R251	FB	24	0
BRGHR	Baud Rate Generator Divisor Register (High)	R252	FC	24	XXXXXXXX
BRGLR	Baud Rate Generator Divisor Register (Low)	R253	FD	24	XXXXXXXX

Table 12. SCI Initialization

Mnemonic	Name	Register	Hex.	Page	Reset value (binary)
RDCPR	Receiver DMA Transaction Counter Register	R240	F0	24	XXXXXXXX
RDAPR	Receiver DMA Address Pointer Register	R241	F1	24	XXXXXXXX
TDCPR	Transmit DMA Transaction Counter Register	R242	F2	24	XXXXXXXX
TDAPR	Transmit DMA Address Pointer Register	R243	F3	24	XXXXXXXX
ACR	Address Compare Register	R245	F5	24	XXXXXXXX
RXBR	Receive Buffer Register (Read only)	R248	F8	24	XXXXXXXX
TXBR	Transmitter Buffer Register (Write only)	R248	F8	24	XXXXXXXX

Table 13. Watchdog/timer configuration/initialization

Mnemonic	Name	Register	Hex.	Page	Reset value (binary)
EIPR	External Interrupt Pending Register	R243	F3	0	0
EIMR	External Interrupt Masking Register	R244	F4	0	0
EIPLR	External Interrupt Priority Register	R245	F5	0	11111111
EIVR	External Interrupt Vector Register	R246	F6	0	XXXX0010
WDTLR	Watchdog Timer Low Register	R248	F8	0	XXXXXXX X
WDTHR	Watchdog Timer High Register	R249	F9	0	XXXXXXX X
WDTPR	Watchdog Timer Prescaler Register	R250	FA	0	XXXXXXX X
WDTCR	Watchdog Timer Control Register	R251	FB	0	10010
WCR	Wait Control Register	R252	FC	0	1111111

Table 14. SPI initialization

Mnemonic	Name	Register	Hex.	Page	Reset value (binary)
SPIDR	SPI Data Register	R253	FD	0	XXXXXXXX
SPICR	SPI Control Register	R244	F4	0	100000

Table 15. EEPROM initialization (ST9040 only)

Mnemonic	Name	Register	Hex.	Page	Reset value (binary)
EECR	EEPROM Control Register	R241	F1	0	0

Appendix B Examples of ST9 peripheral configurations

```
            . sbttl " ST9030 registers addresses and contents "
            . include "c:\st9\bin\symbols.inc"
            ; The reader should refer to the file containing the
            ; declaration of all the bits and registers of the ST9030
            ; for the symbols used in the following listing.
            ;
            ; .nlist
;*********************************************************************
;* This program demonstrates the configuration of ST9 peripherals*
;*********************************************************************
;**********************
;*RAM Declaration*
;**********************
prescal_t0 := r2            ; Value of Timer 0 Prescaler
val_capt_t0 := rr4          ; Value of Timer 0 Capture register
nb_event_t0 := rr4          ; Number of Timer 0 event
lg_dma := rr6               ; Length of DMA
CPT_AD_DMA := RR8            ; DMA Address Register
CPT_LG_DMA := RR8           ; DMA Counter Register
ad_conv := r3                ; conversion start address
IT_T0_LEVEL = 4             ; Timer 0 priority level
IT_CAD_LEVEL = 6            ; A/D converter priority level
;****************************
;*INTERRUPT VECTOR ADDRESSES*
;****************************
CORE_IT_VECT := 00h         ; Core interrupt vectors
T0_IT_VECT := 10h           ; Timer 0 interrupt vectors
EXT_IT_VECT := 20h          ; External interrupt vectors
ADC_IT_VECT := 30h          ; A/D Converter interrupt vectors
SCI_IT := 40h               ; SCI interrupt vector
;*******************
;*STACK Declaration*
;*******************
SSTACK := 223               ; System stack address group D C
USTACK := 191                ; User stack address group B

;*******************
```

```
;*Group number names*
;********************
BK0 := 0
BK1 := 1
BK2 := 2
BK3 := 3
BK4 := 4
BK5 := 5
BK6 := 6
BK7 := 7
BK8 := 8
BK9 := 9
BKA := 10
BKB := 11
BKC := 12
BKD := 13
BKE := 14
BKF := 15
BK_0 := BK0 * 2             ; free user group
BK_BDT:= BK2 * 2            ; TWD group
BK_CAD:= BK5 * 2            ; A/D group
BK_T0 := BK4 * 2             ; MFTimer 0 group
BK_SCI:= BK6 * 2            ; SCI group.
BK_F := BKF * 2             ; paged registers


;****************************************
;*Declaration of the interrupt vector table*
;****************************************
.text                       ; start of program
          .org CORE_IT_VECT ; Core interrupt vector
                            *********************
          .word DIV0        ; divide by 0 interrupt vector
          .word TOP_LEVEL_IT; Top level interrupt vector
          .org T0_IT_VECT   ; Timer 0 interrupt vector
          ; ***********************
          .org T0_IT_VECT + 4 ; unused addresses
          .word T0_CAP       ; Timer 0 capture interrupt vector
          .word T0_COMP      ; Timer 0 compare interrupt vector
```

```
          .org EXT_IT_VECT   ; External interrupt vector
                             ; *************************
WDT_IT:   .word TEMPO ; Watchdog Timer interrupt vector
          .org ADC_IT_VECT  ; ADC interrupt vector
                             ; ********************
          .word RESET_START ; Analog Watchdog interrupt vector
          .word ADC_EOC ; End of conv. interrupt vector
          .org SCI_IT       ; SCI interrupt vector
          ; ********************
          .org SCI_IT + 4   ; unused addresses
          .word REC_DATA    ; receiver interrupt
          .word TRA_HOLD     ; Transmitter interrupt


;**********************
;*Start of main module*
;**********************
          .org 100h          ; start of code
RESET_START:
          ld MODER,#11100000b ; CLOCK MODE REGISTER
                             ; internal stack
                             ; no prescaling
                             ; external clock divided by 2
          ld CICR,#10000111b ; CENTRAL INTERRUPT
                             ; CONTROL REGISTER
                             ; priority level = 7
                             ; concurrent mode
                             ; disable interrupt
          clr FLAGR
          spp #WDT_PG
          ld WCR,#wden       ; watch dog mode disabled,
                             ; no wait states.
          ld EIMR,#0          ; mask all channel interrupts.
                             ; at reset,Global Counter Enable
                             ; bit is active.
          ld SSPLR,#SSTACK + 1 ; load system stack pointer
          ld USPLR,#USTACK + 1 ; load user stack pointer
          call INIT_IO       ; init I/O ports
MAIN:
          jxt MAIN           ; include your Main program here !
```

```
;*************************************************************
;*Configuration of TIMER 0 I/O pins and A/D Converter I/O pins*
proc INIT_IO [PPR] {
;.............
;.............. P3.0 (T0INA) P3.2 (T0INB) INPUT TRISTATE TTL
;.............. P3.1 (T0OUTA) P3.3 (T0OUTB) OUTPUT ALTERNATE FUNCTION
                            PUSH_PULL TTL
          spp #P3C_PG       ; Port 3 control register page
          ld P3C0R,#00001111b
          ld P3C1R,#00001010b
          ld P3C2R,#00000101b
;.............. end of init. P3
;.............. INITIALIZATION OF A/D CONVERTOR INPUTS
;.............. P4.7 (AIN7) ALTERNATE FUNCTION OPEN DRAIN TTL
;.............. P4.6 (AIN6) ALTERNATE FUNCTION OPEN DRAIN TTL
          spp #P4C_PG       ; Port 4 control register page
          ld P4C0R,#11000000b
          ld P4C1R,#11000000b
          ld P4C2R,#11000000b
;............. end of init. P4
;............. INITIALIZATION OF SCI I/O
                            ; P70: Input = Sin.
                            ; P71: AF = Sout.
                            ; P72: AF = Txclck.
                            ; P73: AF = Rxclck.
          spp #P7C_PG        ; Port 7 control page.
          ld P7C0R,#00001111b ; bit 0 (Sin): IN, TRI, TTL.
          ld P7C1R,#11111110b ; bit 1,2,3 (Sout, Txck, Rxck): AF,PP,TTL.
          ld P7C2R,#00000001b ; Others : OUT,PP,TTL.
          }



;*****************************************
;*SECTION CODE FOR THE CORE INTERRUPT ROUTINE*
;*****************************************
;_____________________
```

```
;*INTERRUPT ROUTINE FOR ZERO DIVISION*
;_________________
DIV0:
          nop
          ret
;_________________________
;*INTERRUPT ROUTINE FOR TOP_LEVEL_IT*
;_________________________
TOP_LEVEL_IT:
          nop
          iret
;____________________________
;*INTERRUPT ROUTINE FOR TIMER WATCHDOG INT*
;____________________________
TEMPO:
          nop
          iret
```

Appendix C Examples of timer 0 configurations

```
;***********************
;*DEFINE TIMER 0 MACROS*
;***********************
.macroT0_START_IT           ; start timer 0, enable interrupts
          spp #T0D_PG       ; select Timer 0 data register page
          and T_TCR,#ccl    ; counter clear bit
          or T_TCR,#cen      ; counter enable bit
          or T_IDMR,#gtien   ; global interrupt mask
.endm
.macroT0_START_DMA_CAP      ; start timer 0, enable interrupts
                            ; and DMA
          spp #T0D_PG       ; select Timer 0 data register page
          or T_IDMR,#( gtien | cp0d ); global interrupt mask
          or T_TCR,#cen     ; counter enable bit
.endm
.macroSTOP_T0               ; stop Timer 0
          spp #T0D_PG        ; select Timer 0 data register page
          and T_IDMR,#gtien ; global interrupt mask
          and T_TCR,#cen     ; counter enable bit
.endm

;************************************************************
proc GEST_T0_ITCAPT{
          ;Configuration of Timer 0 for IT CAPTURE
          ;TCR:             - stop count
          ;                 - clear on capture
          ;                 - up count
          ;TMR:             - disable output
          ;                 - internal clock
          ;                 - disable bivalue mode
          ;                 - disable retrigger mode
          ;                 - disable REG1 mode
          ;                 - continuous mode
          ;                 - enable REG0 mode
          ;ICR:             - EXTA Trigger
          ;                 - falling edge on EXTA
```

```
;                   - EXTB No Operation
;OACR-OBCR:         - no operation
;IDMR:              - Interrupt on capture REG0
;DCPR:              - reset value
;DAPR:              - 00h
;IVR:               - Interrupt vector 10h = T0_IT_VECT
;IDCR:              - level 4
spp #T0D_PG         ; Timer 0 data register page
ld T_TCR,#01001000b ; TCR
ld T_TMR,#00001010b ; TMR
ld T_ICR,#01010100b ; ICR
ld T_PRSR,prescal_t0 ; PRESCALER
ld T_OACR,#11111100b ; OACR
ld T_OBCR,#11111100b ; OBCR
ld T_FLAGR,#00h ; FLAGR
ld T_IDMR,#00100000b ; IDMR
spp #T0C_PG ; Timer 0 control register page
ld T0_DCPR,#00h ; DCPR
ld T0_DAPR,#0 ; DAPR
ld T0_IVR,#T0_IT_VECT ; IVR interrupt vector 14h
ld T0_IDCR,#IT_T0_LEVEL ; priority level 4
T0_START_IT ; start Timer 0, enable interrupt
}


;*********************************************************************
proc GEST_T0_EVENT{
          ; Configuration of Timer 0 into EVENT COUNTER MODE
          ; IT COMPARE is serviced when nb_event_t0 is reached
          ;TCR:              - Stop count
          ;                  - Up count
          ;                  - Clear on compare
          ;TMR:              - Disable output 0-1
          ;                  - no Bivalue mode
          ;                  - no Bicapture
          ;                  - Internal clock
          ;                  - Disable retrigger mode
          ;                  - Continuous mode
          ;ICR:              - EXTB Ext.Clock
```

```
            ;                   - Falling edge on EXTB
            ;                   - EXTA I/O
            ;OACR-OBCR:         - No operation
            ;FLAG:              - reset value
            ;IDMR:              - IT compare 0
            ;DCPR:              - 00h
            ;DAPR:              - 00h
            ;IVR:               - interrupt vector 10h = T0_IT_VECT
            ;IDCR:              - priority level 4
            ;COMP0
            spp #T0D_PG ; Timer 0 data register page
            ldw T_CMP0R,nb_event_t0 ; COMP0
            ld T_TCR,#00111000b ; TCR
            ld T_TMR,#00000010b ; TMR
            ld T_ICR,#01000010b ; ICR
            ld T_PRSR,prescal_t0 ; PRESCALER
            ld T_OACR,#11111100b ; OACR
            ld T_OBCR,#11111100b ; OBCR
            ld T_IDMR,#00000100b ; IDMR

spp #T0C_PG ; Timer 0 control register page
            ld T0_DCPR,#0 ; DCPR
            ld T0_DAPR,#0 ; DAPR
            ld T0_IVR,#T0_IT_VECT ; IVR
            ld T0_IDCR,#IT_T0_LEVEL ; IDCR
            T0_START_IT
            }
;*********************************************************************
            proc GEST_T0_DMA{
;Configuration of Timer0 in IT CAPTURE associated to the DMA mode
;the length of DMA is given by lg_dma
            ;TCR:               - Stop count
            ;                   - no clear
            ;                   - Up count
            ;TMR:               - disable interrupt
            ;                   - no bivalue mode
            ;                   - no capture
            ;                   - external/internal clock
            ;                   - disable retrigger mode
```

```
                ;                - continuous count
                ;ICR:            - EXTA TRIGGER
                ;                - Falling edge on EXTA
                ;                - EXTA no operation
                ;OACR-OBCR:      - no operation
                ;IDMR:           - no interrupt, DMA / CAPTURE REG0
                ;DCPR:           - DMA ext. data/program memory- DMA counter
                ;DAPR:           - DMA external program memory - DMA address
                ;IVR:            - interrupt vector 10h = T0_IT_VECT
                ;IDCR:            - interrupt dma priority level 4
                spp #T0D_PG      ; select Timer 0 data register
                ld T_TCR,#01001000b ; TCR
                ld T_TMR,#00001010b ; TMR
                ld T_ICR,#01010100b ; ICR
                ld T_PRSR,prescal_t0 ; PRESCALER
                ld T_OACR,#11111100b ; OACR
                ld T_OBCR,#11111100b ; OBCR

                ld T_FLAGR,#00h  ; FLAGR
                ld T_IDMR,#00100000b ; IDMR
                spp #T0C_PG      ; select Timer 0 control register
                ld T0_DCPR,#CPT_LG_DMA ; DCPR lg. DMA = 4ch = rr12
                                 ; = RR76
                ld T0_DAPR,#CPT_AD_DMA ; DAPR ad. DMA = 48h = rr8
                                 ; = RR72
                ld T0_IVR,#T0_IT_VECT ; IVR
                ld T0_IDCR,#IT_T0_LEVEL ; priority level 4
                ldw CPT_LG_DMA,lg_dma ; init DMA counter
                ldw CPT_AD_DMA,#0ff00h ; DMA address in ROM is 0FF00h
                T0_START_DMA_CAP ; enable Interrupt. and DMA
                }
;*********************************************************************
; Example for Timer 0 and Timer 1 in parallel mode
; A Toggle is generated on T0OUTB and T1OUTB on each overflow
;*********************************************************************
                ;******************
                ;initialize TIMER 0
                ;******************
TIMER0::
```

```
          spp #T0D_PG       ; select timer 0 register page
          srp #BK_F         ; select working register
          ld t_tcr,#00011000b ; Counter clear
                            ; Software Up
          ld t_tmr,#10001000b ; Enable output 1
                            ; Disable output 0
                            ; Not bivalue mode
                            ; REG 1 monitor counter value
                            ; REG 0 Capture
                            ; Internal clock
                            ; Retrigger mode
                            ; Continuous mode

          ld t_icr,#00      ; No action on input pins
          ld t_prsr,#00      ; No prescaling
          ld t_oacr,#11111100b ; No action on OUTPUT0
          ld t_obcr,#11110100b ; Toggle on OVF
          ld t_flagr,#00
          ld t_idmr,#00
.macroT0_START ; Start TIMER 0
          spp #T0D_PG       ; select Timer 0 data register page
          or t_tcr,#cen     ; counter enable bit
.endm
;******************
;initialize TIMER 1
;******************
TIMER1::
          spp #T1D_PG ; select timer 1 register page
          srp #BK_F ; select working register
          ld t_tcr,#00011000b ; Counter clear
                            ; Software Up
          ld t_tmr,#10001100b ; Enable output 1
                            ; Disable output 0
                            ; Not bivalue mode
                            ; REG 1 monitor counter value
                            ; REG 0 Capture
                            ; Parallel mode
                            ; Retrigger mode
                            ; Continuous mode
```

```
          ld t_icr,#00      ; No action on input pins
          ld t_prsr,#00     ; No prescaling
          ld t_oacr,#11111100b ; No action on T1OUTA
          ld t_obcr,#11110100b ; Toggle on OVF T1OUTB
          ld t_flagr,#00
          ld t_idmr,#00


.macroT1_START ; Start TIMER 1
          spp #T1D_PG ; select Timer 1 data register page
          and t_tcr,#ccl ; counter clear bit
          or t_tcr,#cen ; counter enable bit
.endm
          or CICR,#10000000b ; Global counter enable
          loop {
          }
;*********************************************************************
; INTERRUPT SUBROUTINES FOR TIMER 0
;*********************************************************************
;These subroutines are serviced on TIMER 0 Interrupts. They come from:
; T0_IT_VECT + 4 for both - IT/CAPTURE
; and - DMA IT/CAPTURE end of block
; T0_IT_VECT + 6 for - IT/COMPARE
;*********************************************************************
; Timer 0 CAPTURE Interrupt subroutine:
; - IT Capture on event on EXTA
; - DMA IT/CAPTURE end of block
T0_CAP:
          spp #T0D_PG       ; Timer 0 data register page
          tm T_FLAGR,#ccp0  ; mask successful capture
          jxz RESET_START   ; this is not an IT CAPTURE
                            ; == Pb
          tm T_FLAGR,#ocp0 ; overrun on Capture 0 ?
          jxnz RESET_START ; yes == RESET
          and T_FLAGR,#~cp0 ; reset successful capture flags
          and T_FLAGR,#~ocp0 ; reset overrun on capture 0 flag
          iret              ; return from interrupt

;*********************************************************************
```

```
;Timer 0 COMPARE interrupt subroutine:
; - IT / COMPARE
T0_COMP:
          spp #T0D_PG ; Timer 0 data register page
          tm T_FLAGR,#cm0 ; mask successful compare
          jxz RESET_START ; RESET if it is not
                          ; an IT COMPARE
          tm T_FLAGR,#ocm0 ; overrun on Compare 0 ?
          jxnz RESET_START ; yes == RESET
          and T_FLAGR,#~cm0 ; reset successful compare bit
          and T_FLAGR,#~ocm0 ; reset overrun compare 0 bit
          iret              ; return from interrupt
;******** END OF TIMER 0 CONFIGURATION EXAMPLES ************
```

Appendix D Examples of A/D converter configurations

```
;*********************************************************************
proc SG_CONV{
; A/D Converter is configured as follows:
;         - one shot conversion
;         - power up mode
;         - IT upon End of Conversion
;         - Start mode
;         - Autoscan from channel number AD_CONV
;         - No INT upon Analog Compare
          spp #AD0_PG       ; A/D converter register page
          ld AD_CLR,#00000100b ; Control logic register
                            ; power up
                            ; Stop
                            ; Single mode
                            ; Channel 0
          ld AD_CRR,#00h    ; Compare result register
          ld AD_ICR,#00100000b ; Interrupt control register
                            ; mask analog watchdog
                            ; enable end of conversion
          or AD_ICR,#IT_CAD_LEVEL ; Priority level = 6
          ld AD_IVR,#ADC_IT_VECT ; Interrupt vector register
          ld r0,ad_conv ; AD_CONV = channel number
          swap r0
          rcf
          rlc r0 ; mask for channel number
          or AD_CLR,r0 ; start conversion address
          ld R10, #40
          loop [R10] {      ; wait 60ms before start the first
                            ; conversion
          nop
          }
          or AD_CLR,#st ; start conversion
          }
;*************************************************************
; A/D END OF CONVERSION INTERRUPT SUBROUTINE
ADC_EOC:
          spp #AD0_PG ; A/D converter register page
```

```
                    ; converter flags
and AD_ICR,#~(ecv | awd) ; end of conversion pending flag
                    ; analog watch_dog pending flag
and AD_CLR,#~(st | pow ) ; stop converter
                    ; power down mode
iret
```

Appendix E Examples of SCI configurations

```
;***********************
; SCI
;constant declarations.
;***********************
PRIORITY_SCI = 4            ; SCI priority level
DIV_9600 = 78               ; BRG divisor for a 9600 baud clock
                            ; with a 12 MHz system clock.
DIV_4800 = 156              ; To generate a 4800 bds clock.
DIV_2400 = 312              ; To generate a 2400 bds clock.
DIV_1200 = 614              ; To generate a 1200 bds clock.
VC_9600 := 4                ; Character for 9600 bauds.
Return = 00dh
LNG_DMA_SCI := 0Fh          ; DMA length.
DEPART_DMA_SCI := 0A0h      ; Start DMA address .
                            ; BK_DMA_SCI reserved for this.
NUM_TDAP := 6               ; Contains DMA transmit address pointer value.
NUM_TDCP := 7               ; Contains DMA transmit address counter value.
data := r2                  ; data hold register
rec_ptr := rr6
rec_cpt := rr8
;*********************************************************************
; function:
;         - I/O ports initialization.
;         - Speed and frame initialization.
;         - Compare register initialization.
;         - Interrupt and DMA configuration.
;
; Interrupt request:
;         - Receive error.
;         - Receiver data.
;         - end of DMA transmit.
;
; inputs: none
;
; outputs:none
;
```

```
;*********************************************************************
proc INIT_SCI {
;-- Communication format configuration.
;
; Communication format is configured as follows:
;
;         - 8 data bit transmitted or received character.
;         - 1 stop bit included in data format.
;         - Parity even.
;         - 9600 Baud communication rate.
;-- SCI configuration.
;
;         - No address bit included between the parity bit and the stop bit.
;         - Address mode: Address interrupt if character match.
;         - DMA permits transmission from EEPROM memory to serial line.
;         - Receiver data interrupt unmask (to detect a received data item).
;         - Transmitter data interrupt unmask (to detect DMA end of block).
;         - Receiver error interrupt unmask (to detect overrun, parity or framing
error).
          spp #SCI1_PG ; SCI register page.
          srp #BK_F ; To address SCI registers with r.
          ld s_brglr,#00 ; Reset SCI
          ld s_chcr,#( wl8 | sb10 | pen | ep | am )
                           ; 8 data bit.
                           ; 1 stop bit.
                           ; Parity even.
                           ; No address bit.
                           ; AME = 0, AM = 1.
                           ; = IT if character match.
          ld s_ccr,#txclk ; Xmit clock source = BRG.
                           ; Receiver clock source = BRG.
                           ; 16x asynchronous mode.
          ld s_acr,#RETURN ; End Of Command acquisition.


          ;-- Interrupt and DMA configuration.
          ld s_ivr,#SCI_IT ; Interrupt vector register.
          ld s_tdcpr,#NUM_TDCP ; Tx DMA counter in register file.
          ld s_imr,#( rxdi | rxa | rxe )
                           ; Mask Transmitter data interrupt.
```

```
                              ; Unmask Receiver data interrupt.
                              ; Unmask Receiver data error interrupt.
                              ; Unmask Receiver address interrupt.
                              ; Reset of the pending bits.
          ld s_idpr,#PRIORITY_SCI ; Mask transmitter DMA request.
                              ; SCI exeptions priority level.
          ld s_brglr,#DIV_9600 ; BRG divisor for 9600 bauds, start SCI
                              ; !!! with a 24 Mhz external clock,
                              ; !!! or 4800 Bds (12 MHz external clock.)
} ;-- end of proc.
;*********************************************************************
; SYNC_COM:
proc SYNC_COM {
          spp #SCI1_PG
          srp #BK_F
          ld R#NUM_TDAP,#(DEPART_DMA_SCI) ; DMA pointer initialisation.
          ld R#NUM_TDCP,#(LNG_DMA_SCI) ; DMA counter initialisation.
          or s_idpr,#txd
                              ; Unmask transmitter DMA request.
                              ; unmask transmitter data interrupt.
          ld s_imr,#txdi
                              ; Unmask Transmitter data interrupt.
                              ; Mask Receiver data interrupt.
                              ; Mask Receiver data error interrupt.
} ;-- End of proc.


*********************************************************************
; REC_DATA: Receive interrupt.
REC_DATA:
          pushu PPR           ; save page pointer.
          pushuw RPP          ; save register pointer pair.
          spp #SCI1_PG        ; SCI register page.
          srp #BK_SCI         ; 16 registers reserved for SCI.
          ld data,S_RXBR      ; Read the data received.
          and data,#07Fh      ; Mask the parity bit.
          ld rec_ptr(rec_cpt),data ; Storage of the received data.
          incw rec_cpt
          cpw rec_cpt,#7      ; End of the table.
```

```
          and S_ISR,#~rxdp  ; Reset receiver data pending flag.
          popuw RPP         ; restore register pointer pair
          popu PPR          ; restore page pointer
          iret


;*************************************************************************
; TRA_HOLD: End of DMA transmitter Interrupt
; Function:
;         - Check Interrupt source.
;         - Disable DMA mask .
;         - Enable Receiver interrupt mask.
TRA_HOLD:
          pushu PPR ; save page pointer.
          pushuw RPP ; save register pointer pair.
          spp #SCI1_PG ; SCI register page.
          srp #BK_F ; To address SCI registers with r.
          tm s_imr,#txeob
          if [SETZ] { ; If a Transmitter End Of Block interrupt.
                          bres S_txeob ; Dis. Transmit end of block pending bit.
                          bres S_txhem ; Reset transmit holding reg. empty .
          ld s_imr,#~( rxdi | rxe)
                          ; Unmask Receiver data interrupt.
                          ; Unmask Receiver data error interrupt.
                          ; Mask Transmitter data interrupt.
          } else {
                          jx RESET_START ; If not a normal interrupt source.
          } ;-- end of if.
          popuw RPP ; restore register pointer pair
          popu PPR ; restore page pointer
          iret
```

Appendix F Examples of watchdog/timer configurations

```
;*********************************************************************
;INIT_WDT: This procedure initializes and starts Watchdog Timer.
;
; Watchdog mode is disabled.
; Timer will down count in continuous mode.
; It will generate an interrupt on channel A0 at each End Of Count.
; -- See the external interrupt parameters initialization.
;*********************************************************************
proc INIT_WDT {
          spp #WDT_PG       ; To access in paged registers with r.
          ld wcr,#wden      ; watch dog mode dis., no wait states.
          clr wdtpr         ; 333 ns(sys.clock=12 MHz) min. count,
                            ; prescaler = 0.
          ldw WDTR,#3003    ; (3003 X 333) ns = 1 ms.
          or wdtcr,#stsp    ; Timer starts down counting.
                            ; Continuous mode.
                            ; Watch Dog disabled.
                            ; Input section disabled.
                            ; Output disabled.
                            ; Interrupt A0 on Timer EOC.
                            ; Top Level Interrupt on SW TRAP.
};-- End of proc.
;***************************************
;*Interrupt on channel A0 initialization*
;***************************************
          spp #WDT_PG
          srp #BK_F         ; page 0 reg. direct addressing mode.
          clr eipr          ; Dis. all external int. pending bits.
          nop                ; See WARNING (Tech. manual-Chap. 8).
          ld eivr,#EXT_IT_VECT ; External interrupt vector.
                            ; IAOS - TLIS = 00 = ...
                            ; ... A0 int. will be on WDT End Of Count.
          ld eiplr,#0FEh
                            ; Priority level: group INTA0,INTA1 = 4,5.
          ld eimr,#ia0sm    ; Unmask Interrupt A0 channel
                            ; (WDT End Of Count).
```

Revision history

Table 16. Document revision history

Date	Revision	Changes
21-Dec-1992	1	Initial release.
02-Nov-2011	2	Updated format and company logo.