AN1083
APPLICATION NOTE

ST72141 BLDC MOTOR CONTROL SOFTWARE AND FLOWCHART EXAMPLE

by Microcontroller Division Applications

INTRODUCTION

The purpose of this application note is to give the flowcharts of the software examples showing how to drive a motor in both current and voltage modes and give examples of Open Loop or Closed Loop speed regulation.

The software examples described in this application note are those generated by the ST7MTC2 Kanda kit for Sensorless mode. 80% of the code is generic, the remaining 20% is specific to the implementation of the ST72141 microcontroller (MCU) in the ST7MTC2 kit (user interface and communication data).

Note: Please refer to Application Note 1321 for details concerning Sensor mode.

The software examples given in the file attached with this application note illustrate Current mode and Closed Loop driving mode for a 4-pole, brushless DC (BLDC) motor.

1 DESCRIPTION OF SOURCE FILES

The source files described in this section refer to the software provided in the file attached with this application note.

– Aut_cc.asm is the main file and also contains the interrupt routines,
– Sub_cc.asm contains the software subroutines,
– Sub_cc.inc contains the subroutine declarations,
– Cst_cc.asm and Cst_cc.inc contain the constant declarations,
– ST72e141.asm and st72e141.inc are the map files.

Paramcc.txt is a complete set of motor parameters. These parameters are declared in the ROM part of the ST72141 MCU (with an include file) and then transferred to the RAM part of the ST72141 MCU at the beginning of the main file. This is because the value of some parameters can be changed while the motor is running. Most of these parameters are coded as indicated in the datasheet.

Deframp.txt is the ramp table for the start-up sequence of the motor. It contains 38 consecutive decreasing step times required to accelerate the motor and to detect the first Back-EMF zero-crossing event.

Def_prep.txt is the current ramp for the pre-positioning phase of the motor. It contains 25 steps of increasing current values required to set the rotor in a given position.

Table of Contents

INTRODUCTION 1

1 DESCRIPTION OF SOURCE FILES 1

2 PERIPHERAL INITIALIZATION AND START-UP SEQUENCE 6

3 MAIN PROGRAM 10

4 THE THREE PRINCIPAL INTERRUPT ROUTINES (C, D AND Z) 12

4.1 FIRST PART OF C AND D INTERRUPT ROUTINES 12

4.2 BODY OF C INTERRUPT ROUTINE 13

4.3 SYNCHRONOUS MODE PART OF C INTERRUPT ROUTINE 14

4.4 FIRST PART OF Z AND R INTERRUPT ROUTINES 16

4.5 BODY OF Z INTERRUPT ROUTINE 17

4.6 TRANSITION FROM SYNCHRONOUS TO AUTO-SWITCHED MODE 18

4.7 AUTO-SWITCHED MODE PART OF C INTERRUPT ROUTINE 19

4.8 LAST PART OF C INTERRUPT ROUTINE 20

5 SPECIAL FEATURES 22

5.1 R INTERRUPT (MTIM TIMER RATIO CHANGE) 22

5.2 SOFTWARE DEMAGNETIZATION 24

5.3 O AND E INTERRUPT ROUTINES 29

6 SUMMARY 30

The Start/Stop order of the motor is given by pin PB0 on the microcontroller board in the Kanda Kit. Do not forget to unplug the communication cable from the microcontroller board to ensure the correct behaviour of pin PB0.

This software (except for the declaration of motor parameters) is for the Current mode and Closed Loop Driving mode in the third stage of the ST7MTC2 Motor Control Evaluation Kit.

Figure 1. BLDC Control Strategy

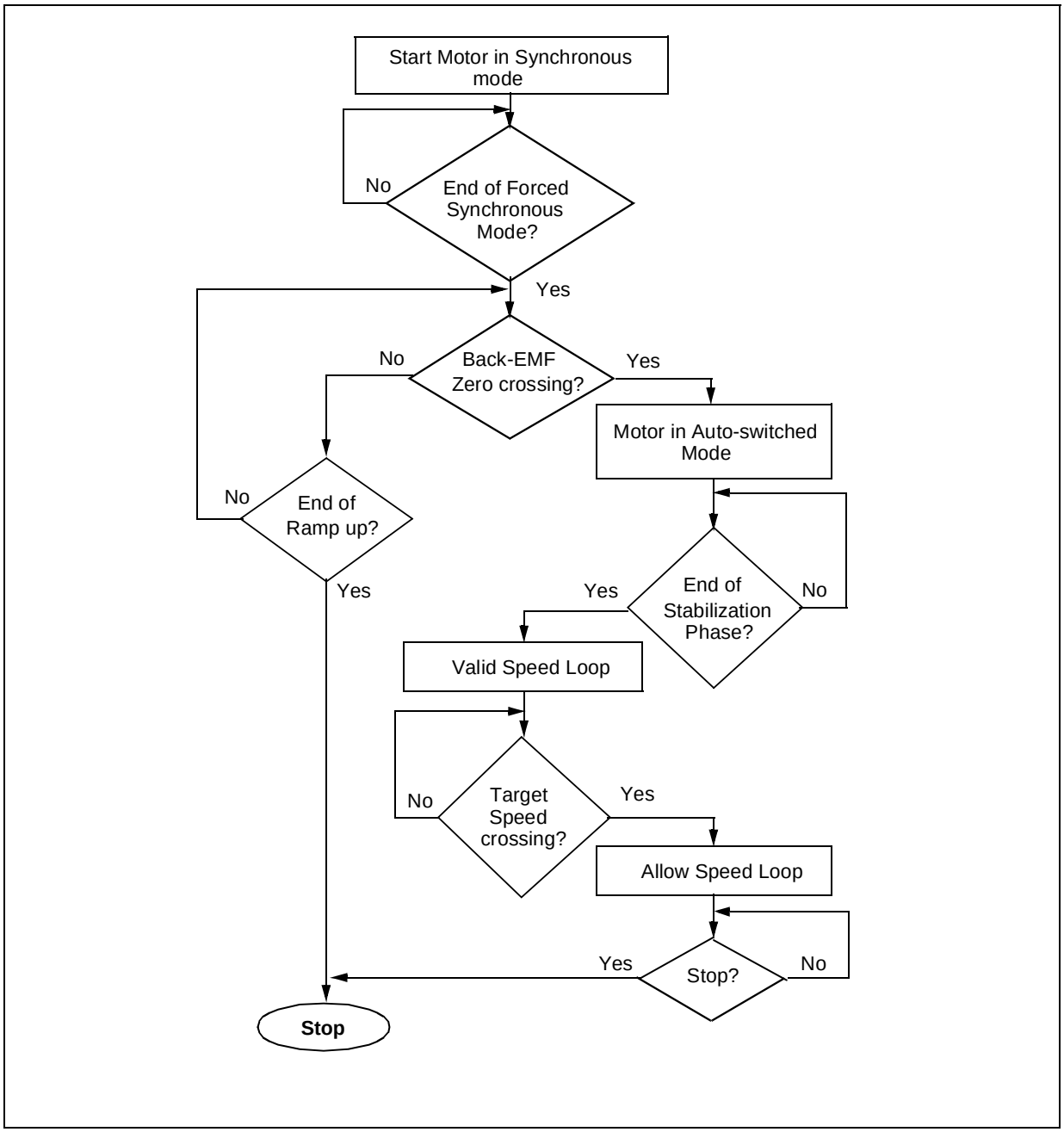


Figure 1 gives a general overview of the control software after receiving an order to start the motor.

After the start order, the motor begins to run in **Forced Synchronous mode**. Forced Synchronous mode is the first part of **Synchronous (switched)** mode. It is the motor start-up sequence. Please refer to Application Note 1276 “BLDC Motor Start Routine for ST72141 Microcontrollers” for details concerning the start-up phase of the motor. During Synchronous (switched) mode, the rotor is set to a given position (pre-positioning phase) and then decreasing step times are imposed in order to accelerate the motor and to detect the first Back-EMF zero crossing event (Z event). The current is also imposed. This succession of decreasing step times is referred to as the motor ramp-up.

Forced Synchronous mode corresponds to the first few steps of the ramp-up before the Z event interrupt is enabled. The number of steps in Forced Synchronous mode (with Z interrupts disabled) is chosen by the software programmer.

After these few steps in Forced Synchronous mode, the Z event interrupt is enabled and the motor is still in the ramp-up, or start-up, phase and switches to Synchronous (switched) mode. The step time and current are still imposed. A Z event has not yet been detected.

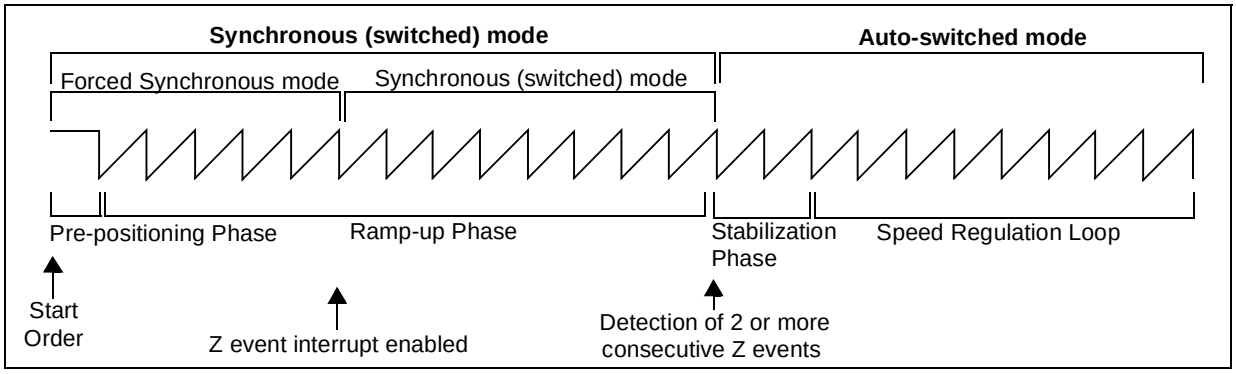
If Z event is not detected before the end of the ramp-up, the motor is stopped.

When the first Z event is detected in Synchronous (switched) mode, the user can choose to have 2 or more consecutive Z events detected before setting the motor in Auto-switched running mode.

Once the motor is in Auto-switched mode, it must wait for the end of the stabilization phase before entering the speed regulation loop. For example, in Closed Loop driving mode, the stabilization phase consists of a few steps between the first step in Auto-switched mode and entering the speed regulation loop. The number of steps in the stabilization phase is chosen by the software programmer.

In summary, after receiving the start order, the motor starts in Forced Synchronous mode, then enters Synchronous (switched) mode when the Z event interrupt is enabled by the software. The motor will then enter Auto-switched mode when 2 or more consecutive Z events are detected.

Figure 2. Start-up Sequence Summary Example



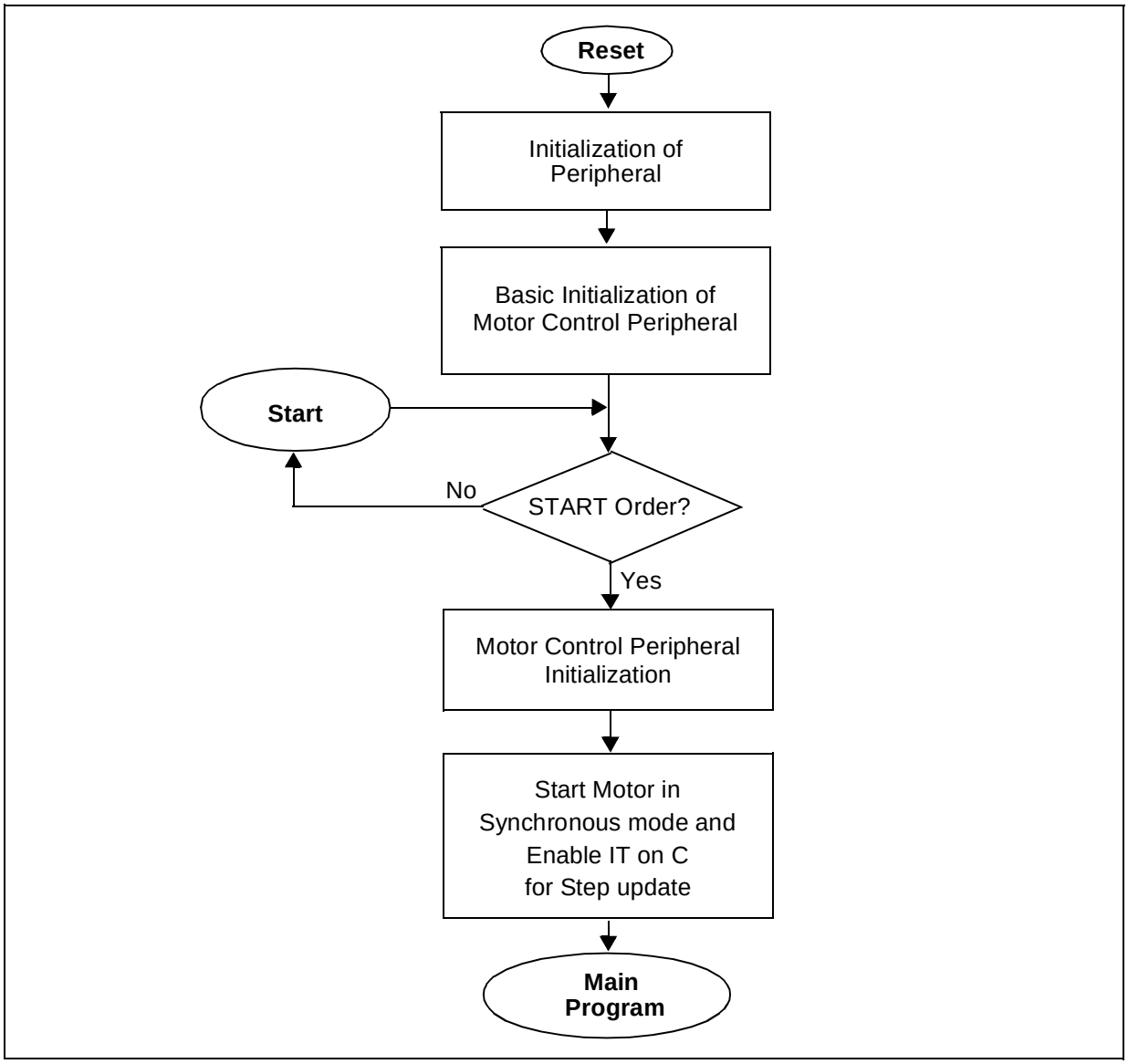
In this application note, the flowchart illustrates the start-up sequence of the motor and the transition between Synchronous and Auto-switched modes. First, the three main interrupt routines are described (C, D, Z events), and then other features are described in Section 5 of this document.

Other special cases are presented at the end of this application note.

Note: All the names of the software routines presented in this document are strictly the same as in the code generated in the ST7MTC2 kit.

2 PERIPHERAL INITIALIZATION AND START-UP SEQUENCE

Figure 3. ST72141 Start-up Sequence Flowchart



Before starting the motor, a basic initialization of the peripheral is performed by the software (Figure 4). Once the peripheral is initialized, the motor waits for the start order.

When the start order is received, the Motor Control (MTC) peripheral is initialized (Figure 5) according to the selected driving mode and motor parameters. The motor then starts in Synchronous (switched) mode and the software switches to the main program (Figure 7).

In **Forced Synchronous mode**, only the C (commutation) event interrupt is enabled. After switching into **Synchronous (switched) mode**, the Z event interrupt is also enabled. The D

(demagnetization) event interrupt can be disabled in both **Forced Synchronous** and **Synchronous (switched) modes**. Otherwise, it is software demagnetization only.

Figure 4. Basic Initialization Phase of the Motor Control Peripheral

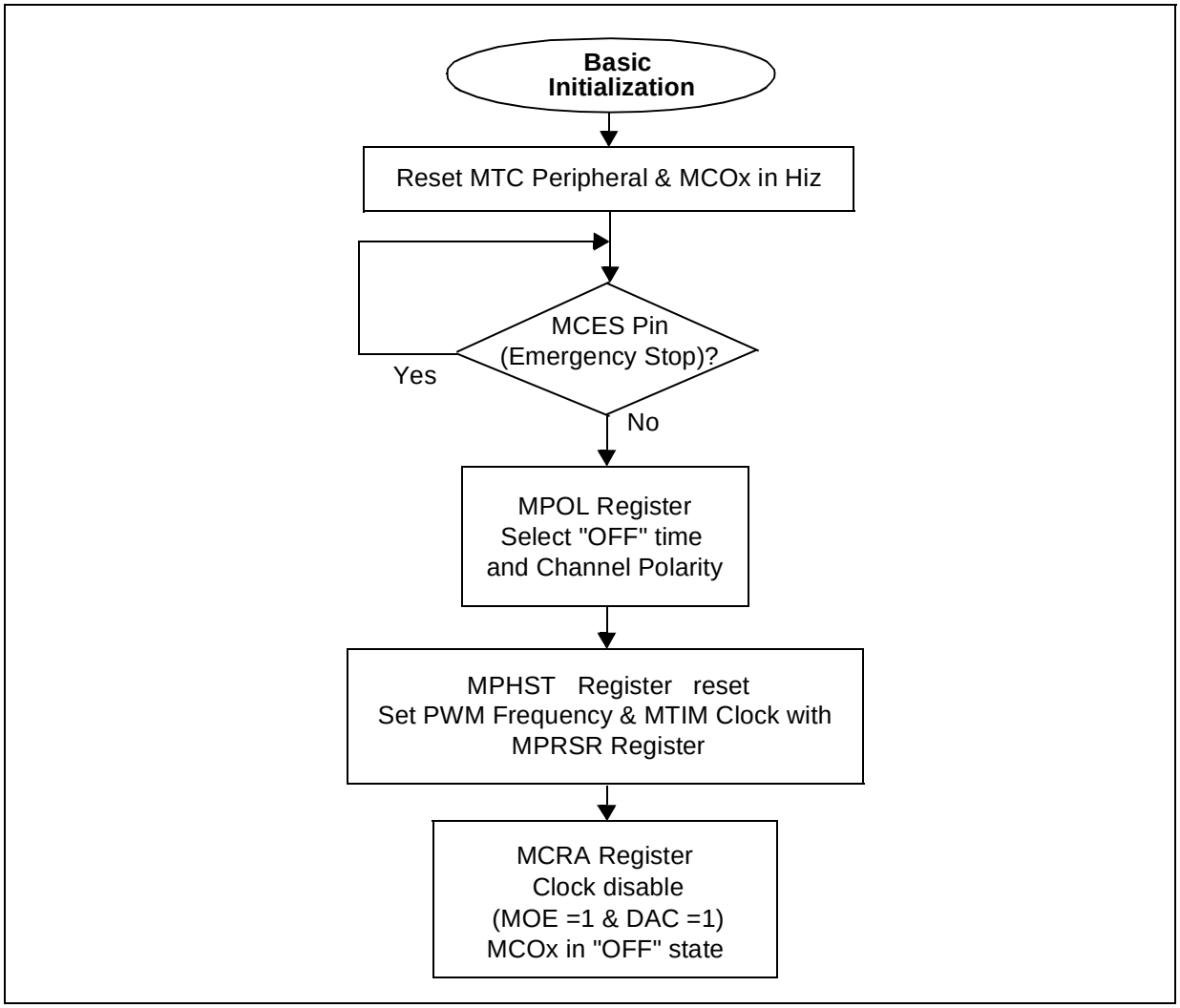


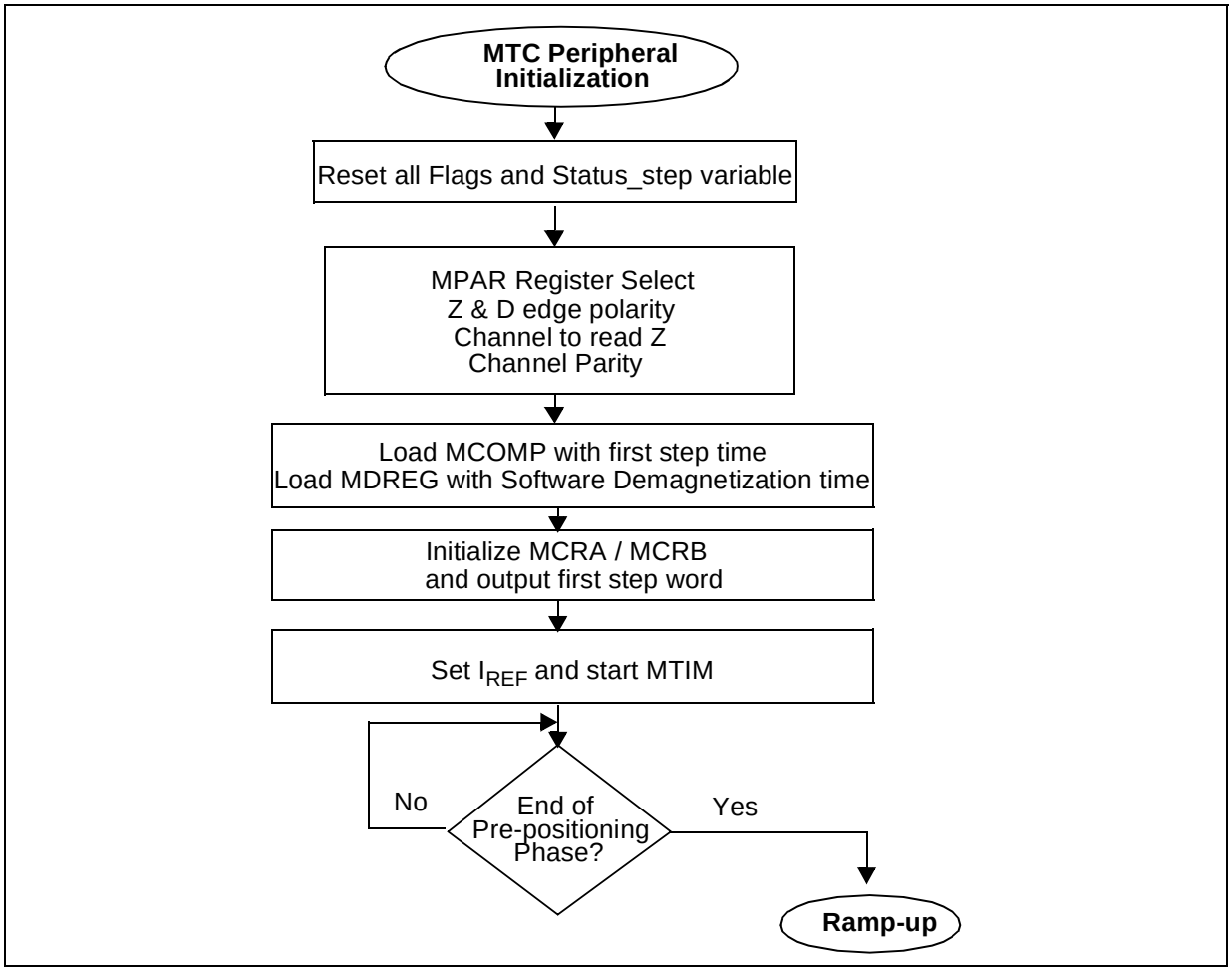
Figure 4 shows the basic initialization phase of the peripheral performed by the software before the detection loop for the motor start-up order can be enabled.

The following actions takes place during the basic initialization phase:

– The Motor Control Peripheral is reset (by setting a control bit in the MCRA register).
– All outputs are set to high impedance and to OFF states.
– The channel polarity is selected (by the MPOL register).
– The internal clock frequency (current mode) is selected as well as the MTIM internal 8-bit timer ratio with the MPRSR register.
– The clock is disabled.

Once the basic initialization is completed and the motor start order is detected, the motor control peripheral is initialized according to the selected driving mode and motor parameters.

Figure 5. Motor Control Peripheral Initialization Phase



When the start order is received:

– All the flags present in the RAM variables are reset, as well as the status_step variable (RAM variable).
– The pre-positioning phase is executed in order to place the rotor in a known position. Refer to Application Note 1276 for more information.
– The motor starts in Synchronous (switched) mode according to the step time in the MCOMP register due to the acceleration ramp table.

The step time is imposed by the ramp table in the software. (The ramp table is the succession of decreasing step times used to accelerate the motor during the start-up sequence). The first step time is loaded in the MCOMP register (in Synchronous mode, the MCOMP register contains the step time).

The MDREG register is loaded with the Software Demagnetization time. In Synchronous (switched) mode, all the Demagnetization (D) events are simulated by software. There are no hardware End of Demagnetization events in Synchronous (switched) mode.

The Z event and D event edge polarities are selected using the ZVD and CPB bits.

The MCRA and MCRB registers are initialized when the driving mode (current or voltage mode with or without sensors) is selected and all the motor parameters are downloaded in the software.

The reference current or voltage for the ramp table is set and the internal 8-bit timer is started by enabling the clock.

In summary, once the Pre-positioning phase is completed, the software goes through the ramp table (ramp-up phase shown in Figure 6) using the decreasing imposed step times to accelerate the motor.

Figure 6. Synchronous Starting Ramp-up

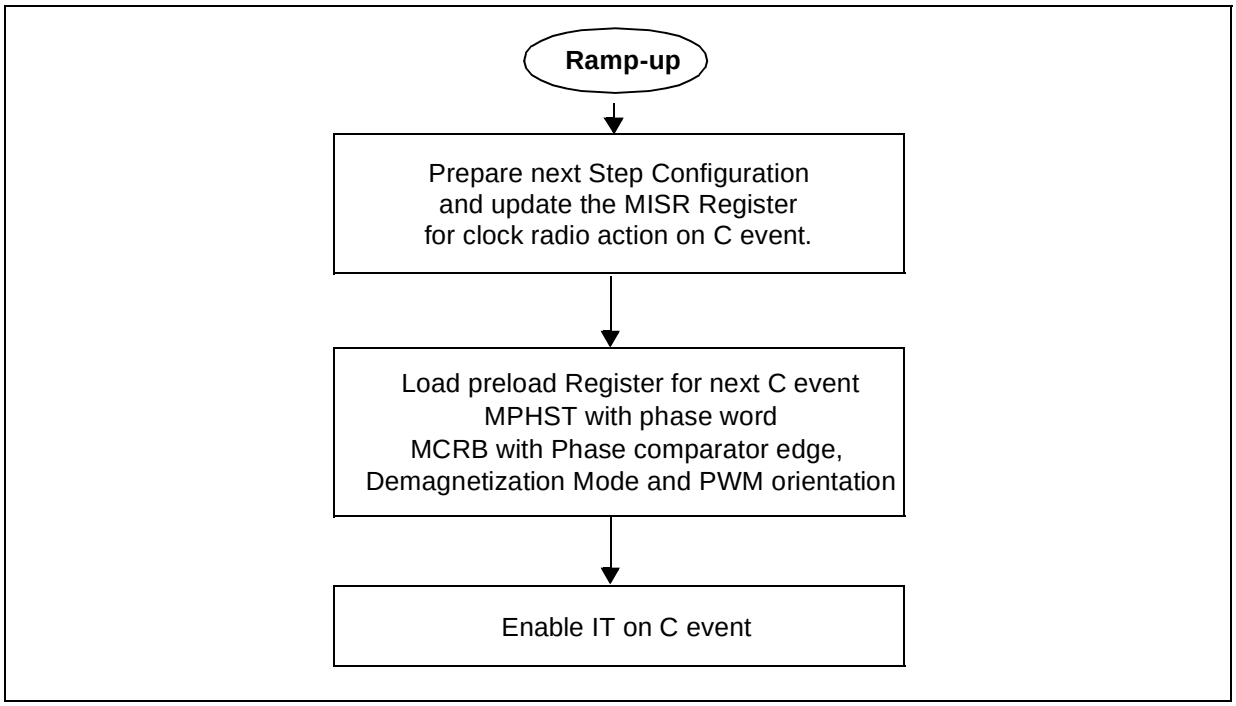


Figure 6 shows what the software does in each step during the ramp up. As the MPHST and MCRB registers have preload registers (see Application Note 1082), in each step the data for the next step is prepared and entered in the preload register. This data will be taken in account and loaded in the active registers when the C event is detected.

In Forced Synchronous mode, only the C event interrupt is enabled. After a few steps in the ramp table, the Z event interrupt will also be enabled.

3 MAIN PROGRAM

Figure 7. Main Program Flowchart

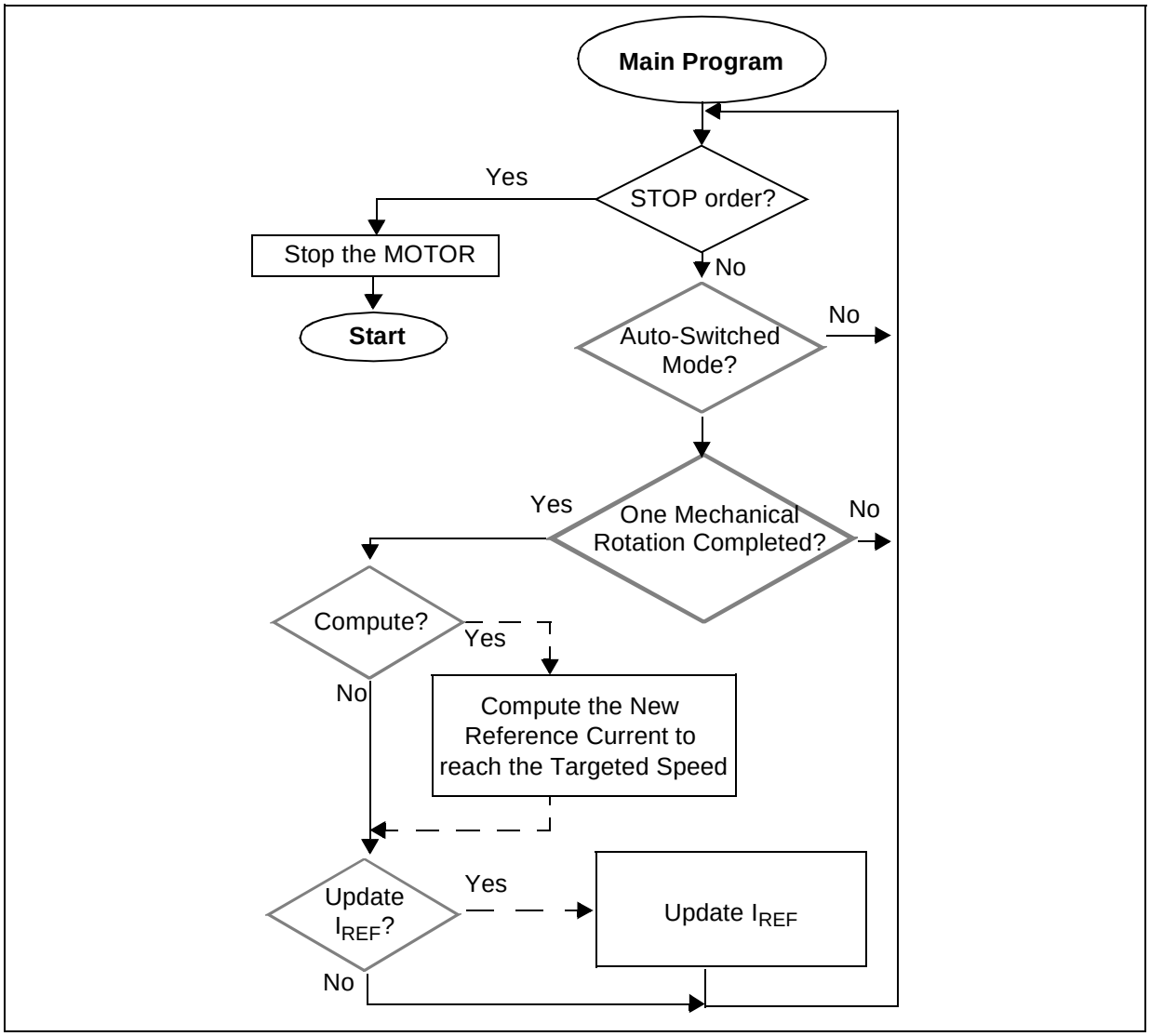


Figure 7 shows what is done in the main program. The main program continuously checks for the stop order. This is the only operation that is carried out until the motor enters the Auto-switched mode. If the motor is in Synchronous (switched) mode, the main program will be interrupted by a C event and a Z event. A D event interrupt is not enabled in Synchronous (switched) mode.

If the motor is in Auto-switched mode, the software waits until one mechanical rotation is completed (this is the stabilization phase, where the number of steps of the stabilization phase is decided by the software programmer). In Auto-switched mode, the main program can be inter-

rupted by all the events enabled by the software programmer. This will be described in Section 5.2 of this document.

The dotted line indicates what is done in the main program in Closed Loop speed regulation mode. After the stabilization phase, the main program checks if there is a need to compute and update a new reference current or voltage in order to keep the motor running at the target speed fixed by the user.

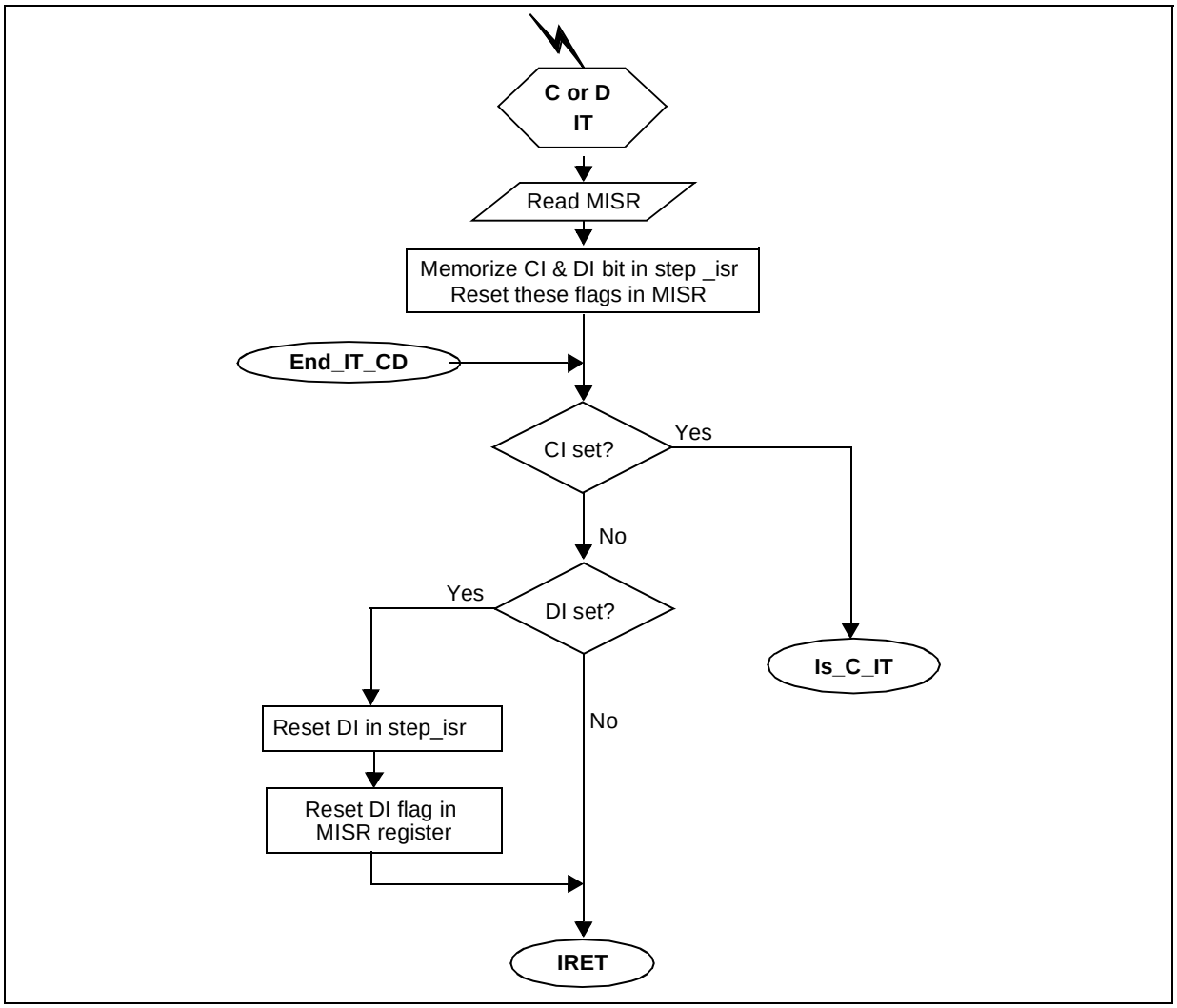
If the motor is still in Synchronous (switched) mode, the software waits for a C event interrupt (Figure 8) before computing the reference current or voltage.

4 THE THREE PRINCIPAL INTERRUPT ROUTINES (C, D AND Z)

4.1 FIRST PART OF C AND D INTERRUPT ROUTINES

When a C event or a D event occurs, the software reads the MISR register and stores the interrupt flags in a RAM variable (step_isr) before resetting these flags in the MISR register.

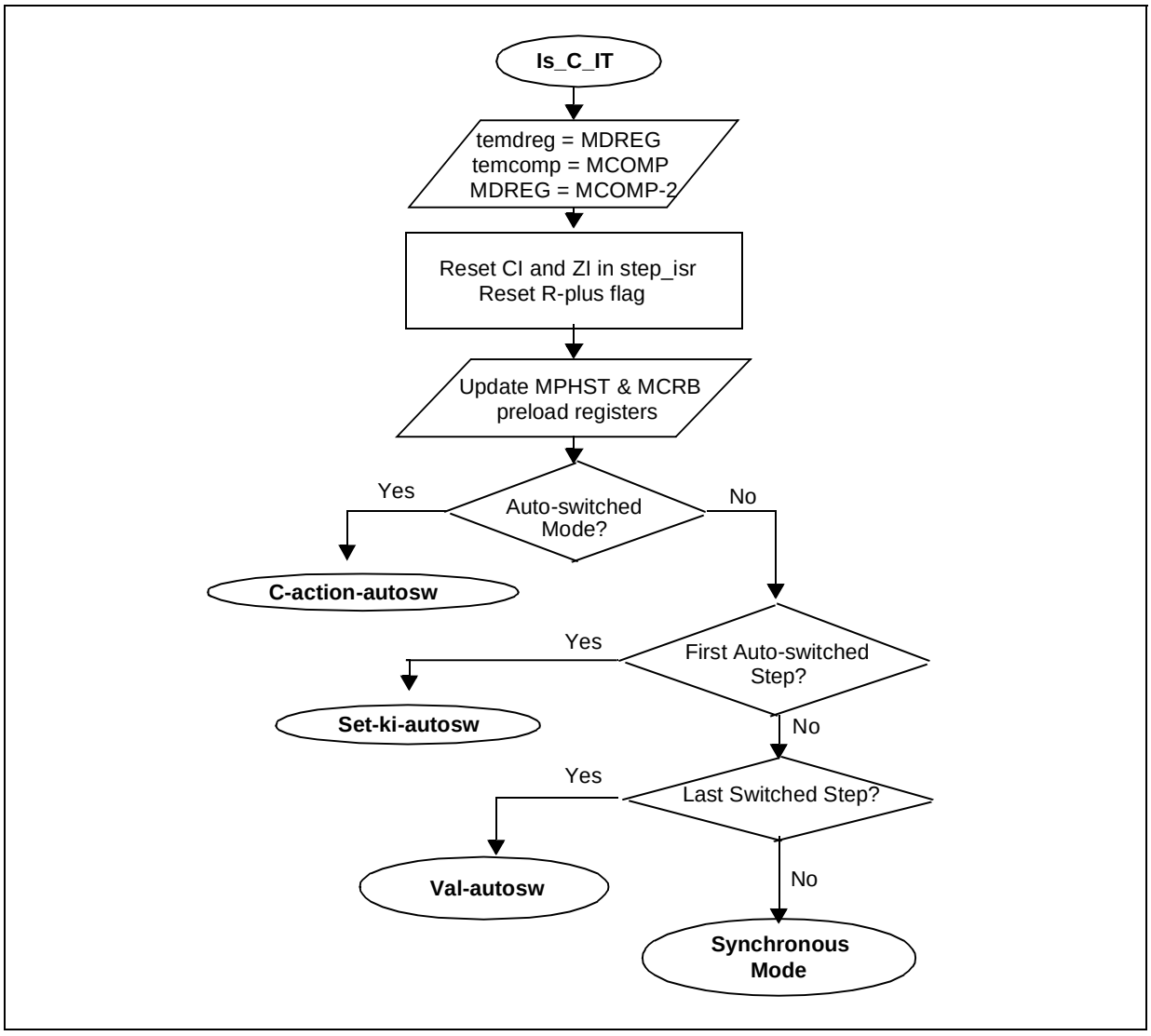
Figure 8. C Event and D Event Interrupt Routines



The RAM variable is then read in order to know what flags have been set. If the commutation flag is set (CI flag), the software enters the C interrupt routine (Is_C_IT shown in Figure 9). If this flag is not set, the software checks if the End of Demagnetization flag is set (DI flag). If the DI flag is set, the software enters the End of Demagnetization event interrupt routine. In this interrupt routine, only the flag reset operation is performed. Afterwards, the software returns to the main program and waits for an interrupt. Figure 9 shows what is done when the commutation event interrupt flag (CI) is set.

4.2 BODY OF C INTERRUPT ROUTINE

Figure 9. Overview of the C Event Interrupt Routine



Regardless of the motor driving mode or sequence during a C event interrupt routine, the Demagnetization time in the MDREG register is stored in a RAM variable. The value of the MCOMP register is also stored. Then, to avoid a parasitic Demagnetization when the interrupt routine is executed, a value less than the value in the MCOMP register is loaded in the MDREG register (meaning that the internal timer has already at that time a value greater than the value loaded in the MDREG register. This will prevent a Demagnetization interrupt from occurring while this part of the interrupt routine is executed).

The interrupt flags are reset and the MCRB and MPHST preload registers are updated to prepare the data for the next C event.

Once the targeted number of consecutive Z events (set by the software programmer) is detected in Synchronous (switched) mode, the fact that the motor is in its last step in Synchronous (switched) mode is memorized (Figure 12).

When the next C event is detected, the software will memorize that the motor is in the first step in Auto-switched mode.

Then the C event interrupt routine is sub-divided to handle the several cases when the interrupt occurs:

– The motor is still in synchronous (switched) mode
– The motor is on the last step in synchronous (switched) mode
– The motor is in the first step in Auto-switched mode
– The motor is already in Auto-switched mode

Let's take each case independently.

In the first case, the motor is in Synchronous (switched) mode when a C event is detected. When this occurs, the software enters the C event interrupt routine and goes to the Synchronous (switched) mode routine (Figure 10).

In the second case, the motor is in its last step in Synchronous (switched) mode (position memorized during the previous Z interrupt routine) when the C event interrupt is detected. When this occurs, the software goes to the val_autosw routine (Figure 13). In this routine, the software memorizes that the motor is in its first step in Auto-switched mode.

In the third case, the motor is in its first step in Auto-switched mode when the C event interrupt is detected. When this occurs, the software goes to the Set_ki_autosw routine (Figure 14).

In the last case, the motor is in Auto-switched mode when the C event interrupt is detected. When this occurs, the software goes to the C_action_autosw routine (Figure 14).

4.3 SYNCHRONOUS MODE PART OF C INTERRUPT ROUTINE

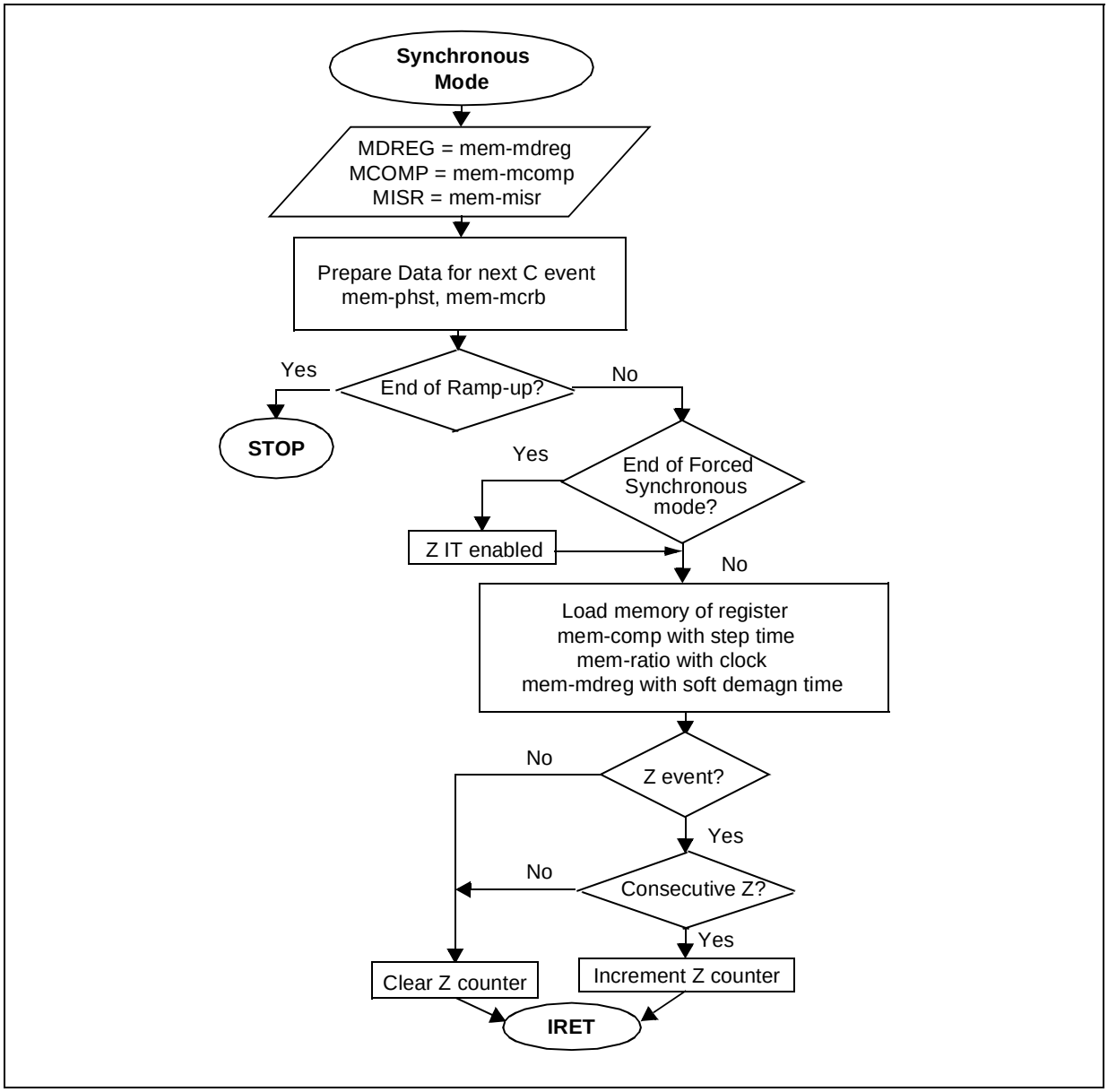
In the event that the motor is still in Synchronous (switched) mode when the C event interrupt occurs, the software goes to the Synchronous mode routine (Figure 10).

When a commutation (C) event occurs, the motor is in Synchronous (switched) mode, so the software must enter the synchronous routine in the C event interrupt routine.

The MDREG, MCOMP and MISR registers are updated for this step. The data for the next C event is prepared and memorized in RAM variables. The data will be loaded in the preload register one step before it is used.

The software checks if the motor is at the end of the ramp table. If this is the case, the motor must be stopped because it is not yet in Auto-switched mode. Normally, the motor enters Auto-switched mode after only a few steps.

Figure 10. Commutation Interrupt in Synchronous (switched) Mode



If the motor is not at the end of the ramp table, the software checks if it is at the end of Forced Synchronous mode. If this is the case, the Z event interrupt is enabled.

All the register values are memorized by the software which also checks if a Z event has occurred in the previous step. When a Z event is detected, a counter is incremented until the number of consecutive Z events reaches the target number set by the software programmer. When this number is reached, the motor switches to Auto-switched mode and the software returns to the main program (Figure 7) and waits for the next interrupt.

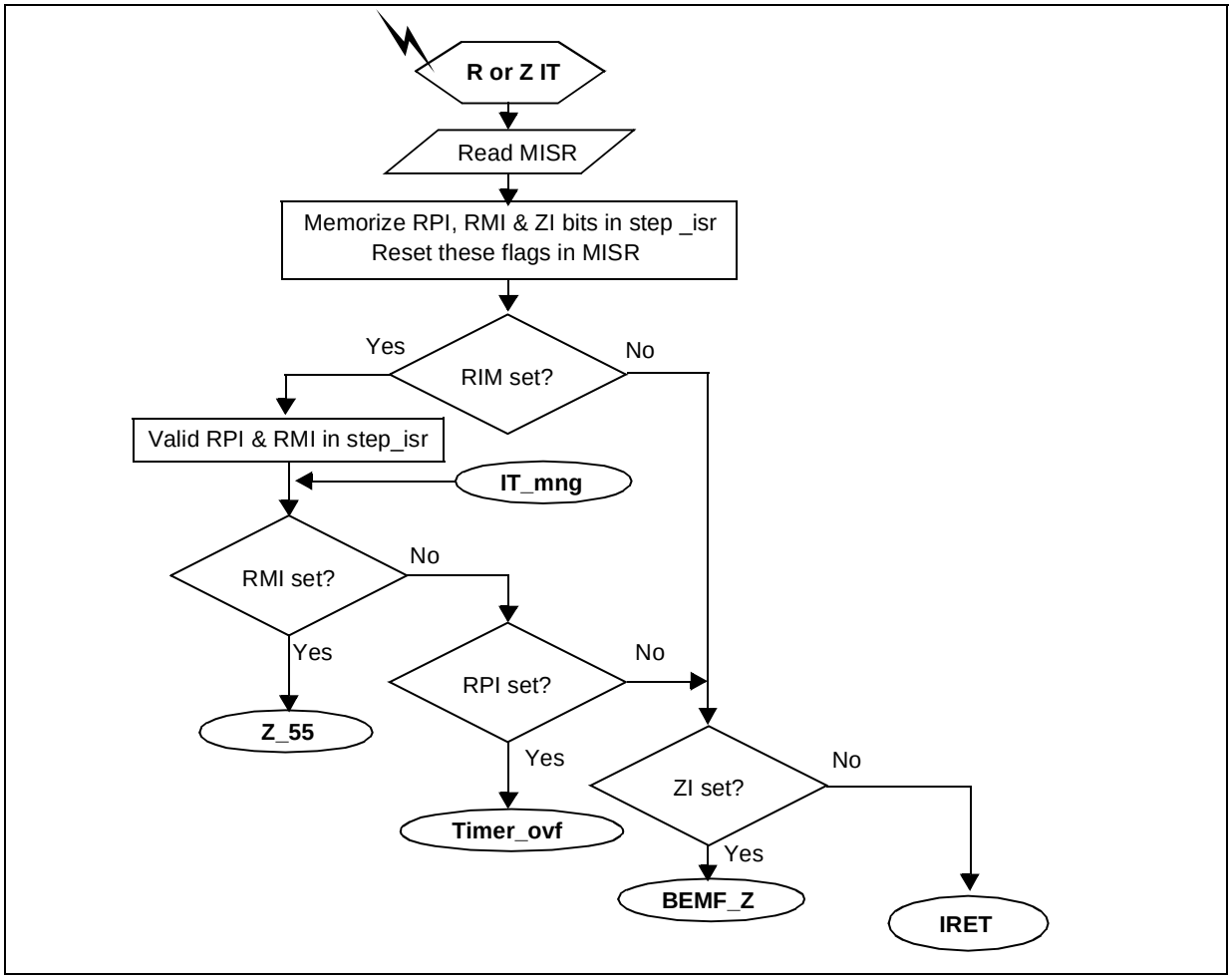
In the event that the motor is in Synchronous (switched) mode and the Z event interrupt has been enabled, the next interrupt in the main program could be a Z event and the software will go through the Z event interrupt routine (Figure 11).

Note that we are assuming for the moment that we are in Synchronous (switched) mode, therefore only the C and Z event interrupts have been enabled by software.

4.4 FIRST PART OF Z AND R INTERRUPT ROUTINES

Figure 11 shows the beginning of the R (Ratio) and Z (Back- EMF zero crossing) interrupt routines. These routines read the interrupt flags, store them in a RAM variable and then reset the flags. If the R event (ratio) flag has been set, the software enters the R event interrupt routine depending on whether it is an R- or an R+ event. This is explained in Section 5 at the end of this application note (Figure 17 and Figure 18). If the Z event flag (ZI) has been set in the MISR register, the software goes through the Z event interrupt routine (BEMF_Z) (Figure 12). Otherwise, the software returns to the main program and waits for the next interrupt.

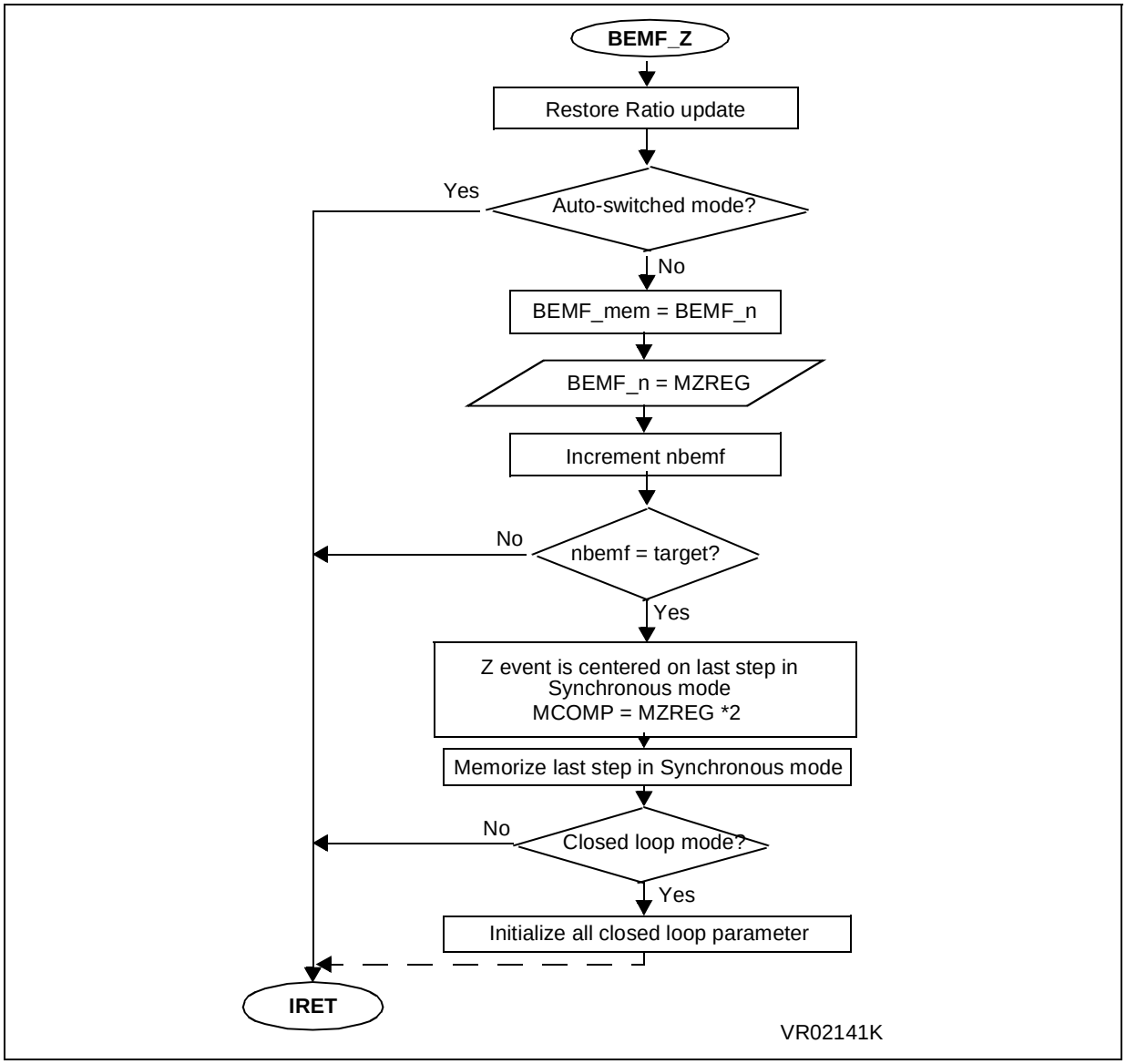
Figure 11. R (Ratio) and Z (Back-EMF) Interrupt Routine



4.5 BODY OF Z INTERRUPT ROUTINE

In Auto-switched mode, no actions are performed in the Z event interrupt routine. An interrupt is generated and the software just returns to the main program (Figure 7). If the motor is still in Synchronous (switched) mode, the zero crossing times (current and previous) are memorized in two RAM variables and the counter of consecutive Z events is incremented. If the number of consecutive zero-crossing events is equal to the target number fixed by the software programmer, the software memorizes that the motor is in its last step in Synchronous (switched) mode.

Figure 12. Z Event Interrupt Routine



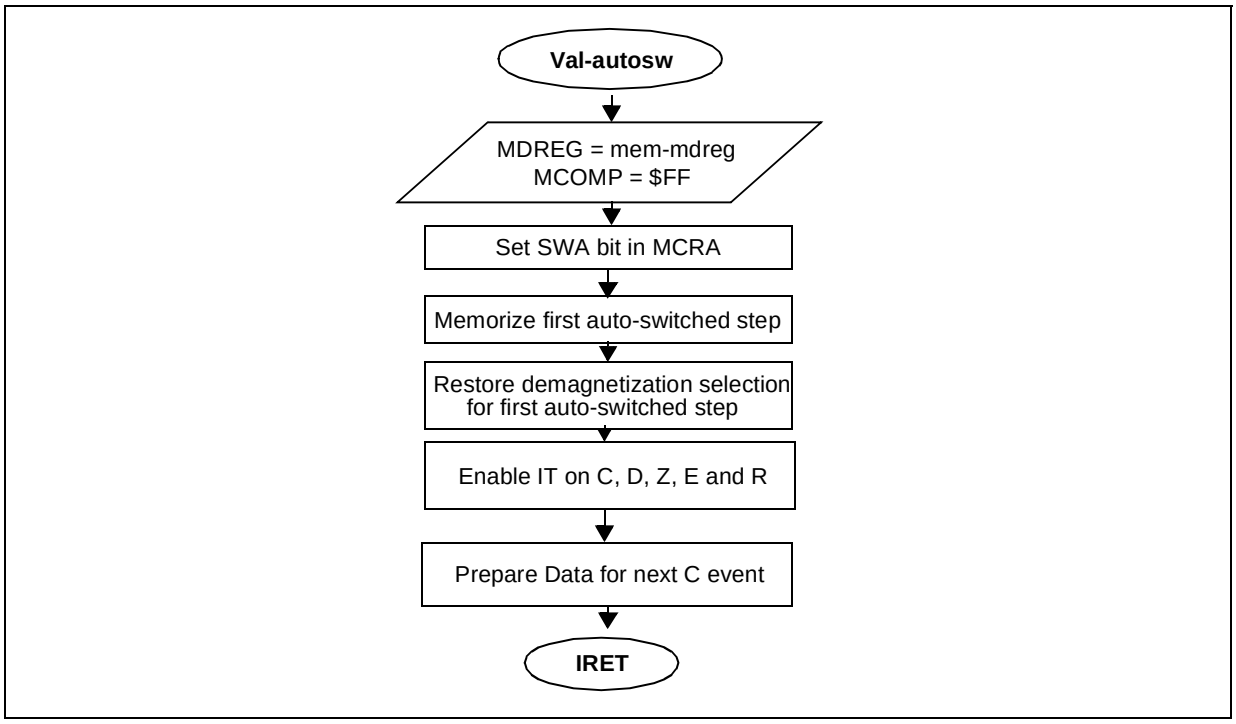
The Z event time is centered on the current step, meaning that the value entered in the MCOMP register (step time in Synchronous mode) is twice the value in the MZREG register.

If the motor is in Closed Loop mode (speed regulation), the Closed Loop parameters are initialized at this point in the program.

The software then returns to the main program (Figure 7) and waits for the next interrupt which will be a C event. As shown in Figure 9, if the software has memorized that the motor is in its last step in Synchronous (switched) mode, the program will enter the Val_autosw routine when the next C event is detected (Figure 13).

4.6 TRANSITION FROM SYNCHRONOUS TO AUTO-SWITCHED MODE

Figure 13. Val_autosw Routine



In this case, the motor is at the last step of the Synchronous (switched) mode and the software has entered the corresponding C event interrupt routine (see Figure 9). Auto-switched mode is enabled and the maximum value is loaded in the MCOMP register to prevent parasitic C events from occurring during the enabling routine.

The SWA bit in the MCRA register is set to enable Auto-switched mode. When this step is completed, the internal 8-bit timer MTIM will be reset on a Z event.

The software memorizes that the motor is in its first step in Auto-switched mode.

The demagnetization selection for the first auto-switched step is restored and all the interrupts are enabled as the motor enters Auto-switched mode.

The E event and R event interrupts are described in Section 5 of this document (see Figure 17, Figure 18 and Figure 22).

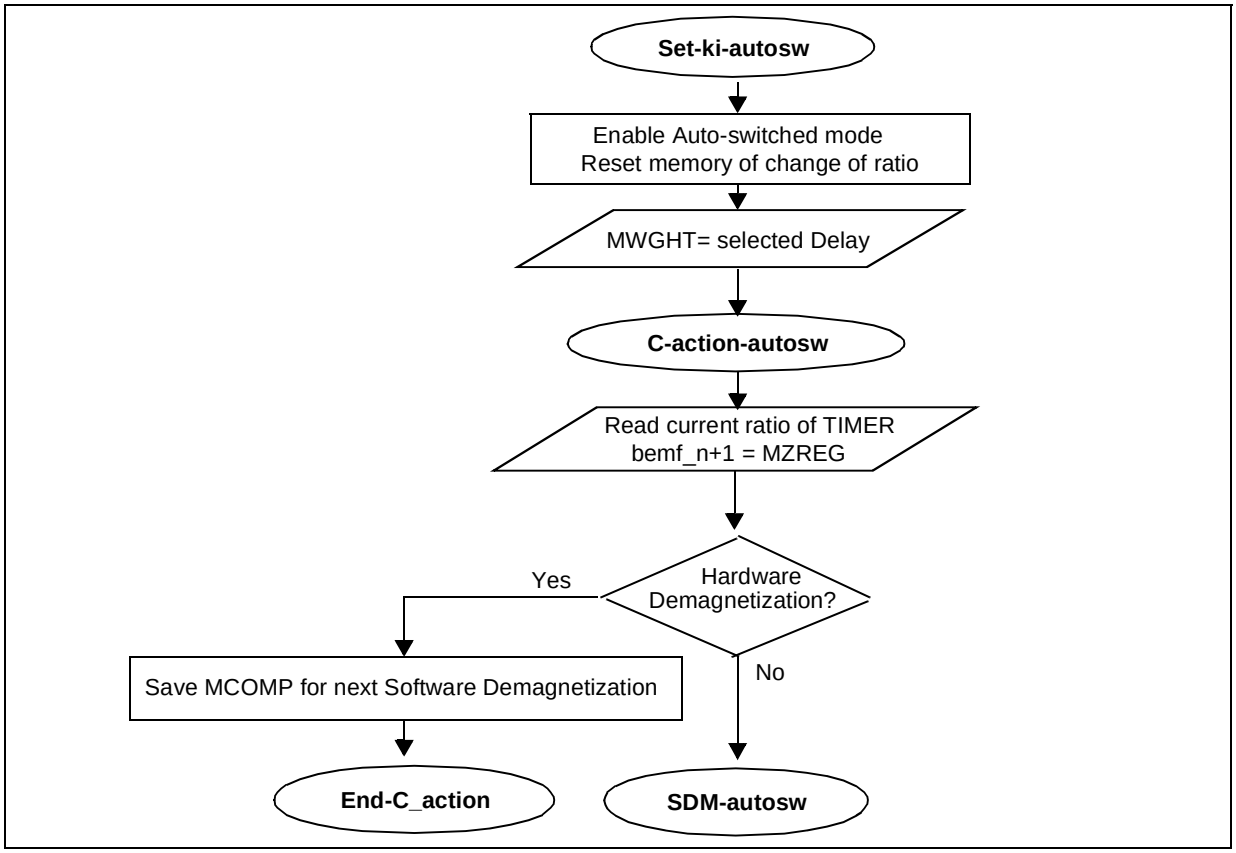
The data for the next commutation event is prepared and memorized so that it may be loaded in the preload registers.

Once this data is stored in the preload registers, the software returns to the main program and waits for the next interrupt which will be a D event. But as shown in Figure 8, when a D event is detected, the interrupt routine only stores the DI bit and clears the MISCR flags before returning to the main program and waiting for the next interrupt.

The next interrupt will be a Z event, but as the motor is now in Auto-switched mode (first step in this mode), this interrupt routine does not perform any actions either (as shown in Figure 12). The software will again return to the main program and wait for the next interrupt which will be a C event. Now, the software has memorized that the motor is in its first step in Auto-switched mode, so as shown in Figure 9, the program will enter the Set_ki_autosw routine (Figure 14).

4.7 AUTO-SWITCHED MODE PART OF C INTERRUPT ROUTINE

Figure 14. Commutation Event Routine (first step in Auto-switched mode)



In this case, Auto-switched mode is enabled and the MWGHT register is loaded with the delay coefficient. This means that the MCOMP register contains the value of the real delay (time in each step between the zero crossing event and the next commutation event).

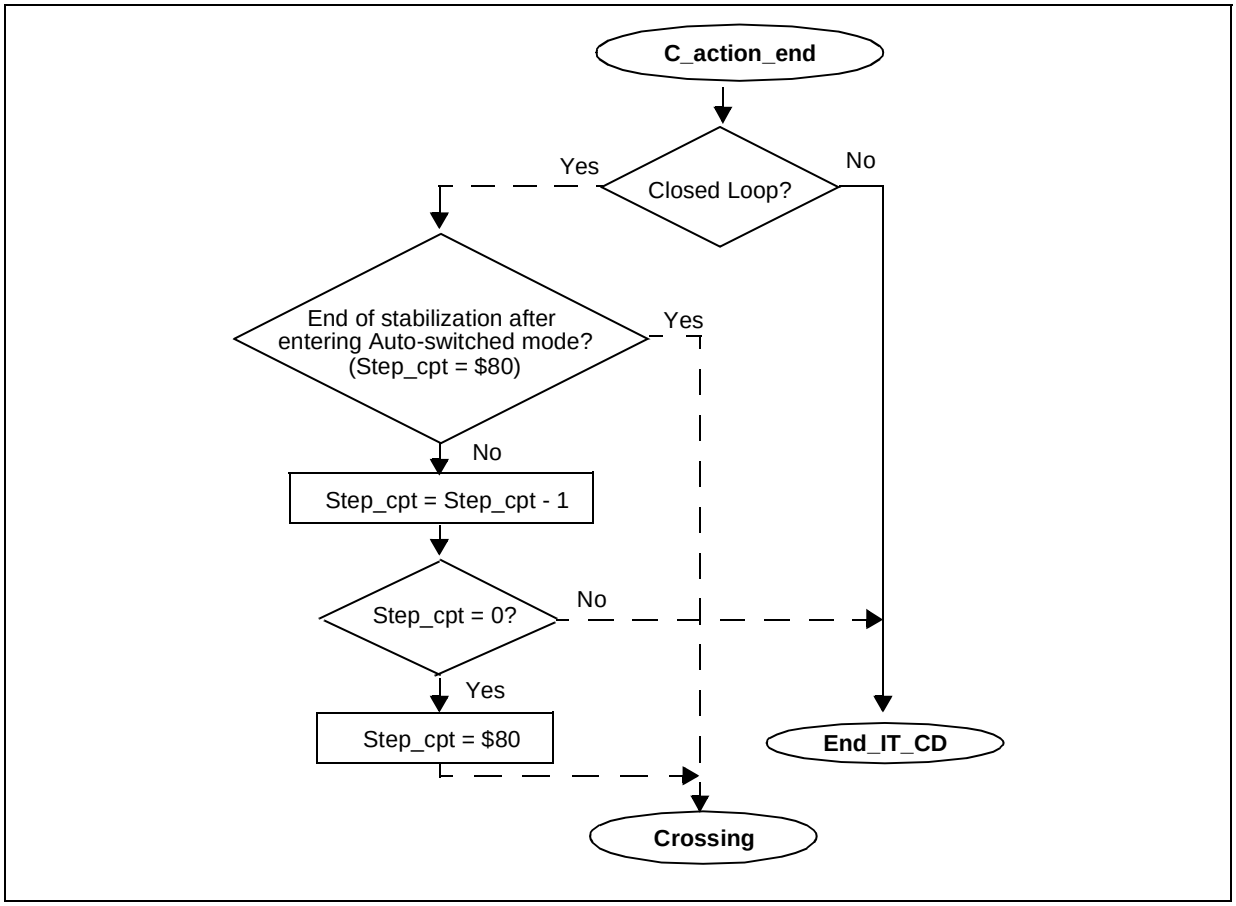
Note: For more information concerning the real delay value, refer to Application Note 1082.

The Z event time is memorized and then depending on the step configuration (falling or rising edge of the end of Demagnetization event), there will be a software or a hardware Demagnetization. Software Demagnetization is a special case that is described in more detail in Section 5.2 of this document (Figure 19 and Figure 20). If there is a hardware Demagnetization, the software enters the End_C_action routine where the data for the next commutation event is prepared. The software then goes to the C_action_end routine (Figure 15).

Note: For more information concerning the step configuration, refer to Application Note 1130.

4.8 LAST PART OF C INTERRUPT ROUTINE

Figure 15. C_action_end Routine



In this case, the execution of the routine depends on whether the motor is running in Open Loop or Closed Loop mode. If the motor is running in Closed Loop (speed regulation), the soft-

ware checks if the stabilization phase is completed (fixed number of steps in Auto-switched mode before entering the speed regulation loop).

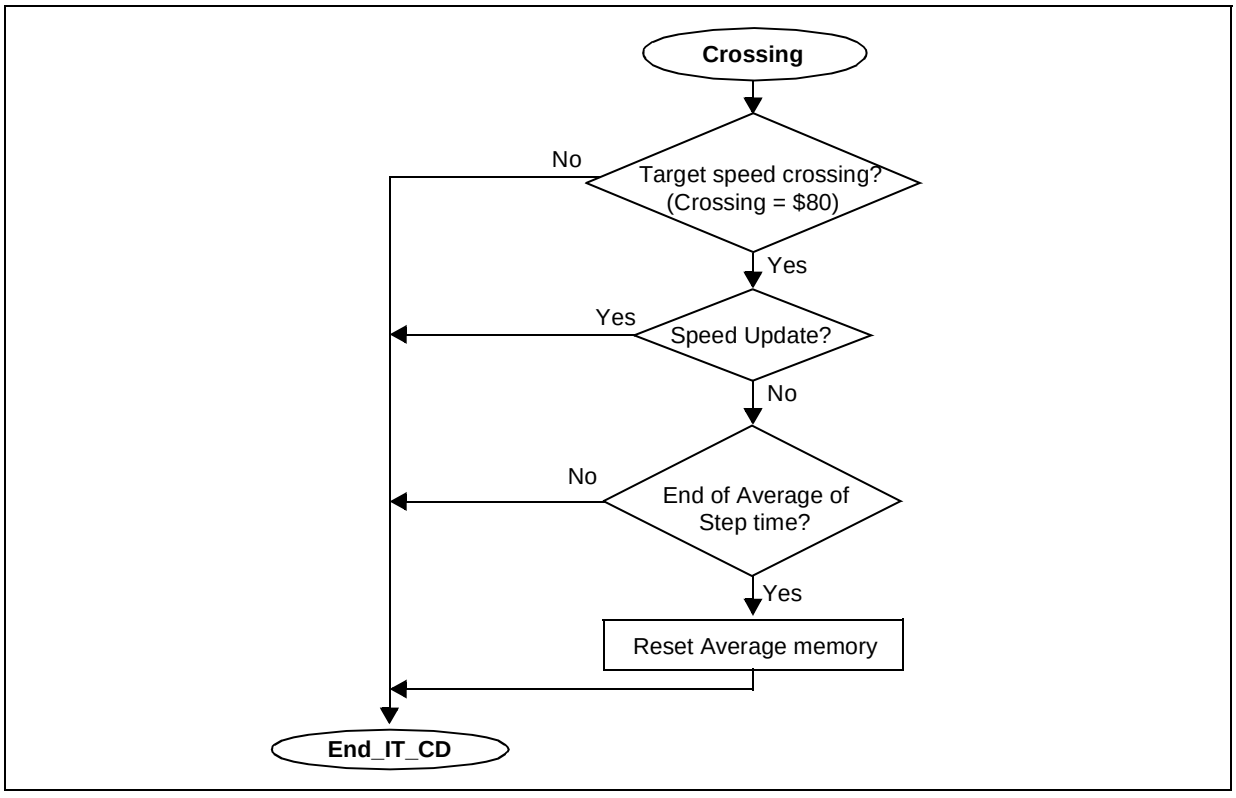
If the stabilization phase is not finished, the value of the counter for the number of steps in Auto-switched mode is incremented and the software goes to the End_IT_CD routine (Figure 8) before returning to the main program.

If the stabilization phase is completed, the software goes to the crossing routine (Figure 16) to check if the target speed of the motor has been crossed.

If the motor is running in Open Loop mode, the software just exits the C interrupt routine.

Note that the number of steps for the stabilization phase is fixed by the software programmer in a RAM variable. Step_cpt acts like a flag and $80 is an arbitrary value set by the software programmer.

Figure 16. Crossing Routine



In this routine, the software checks if the motor has already crossed the target speed. If the motor has not crossed the target speed, the software exits the C interrupt routine. (This routine is only available in Closed Loop mode.)

If the motor has crossed the target speed, the software checks if the speed has to be updated or if the step time average has to be computed by the speed regulation subroutine. Afterwards, the software returns to the main program.

As the motor is now in Auto-switched mode, the same path is always followed by the software:

– Main program: software waits for an interrupt
– C event interrupt is detected: the software enters the C_action_autosw (Figure 14) routine then returns to the main program
– D event interrupt is detected: nothing is done in the software
– Z event interrupt is detected: nothing is done in the software on a Z event in Auto-switched mode

However, C, D and Z events are not the only events that can provoke an interrupt. In Auto-switched mode, R (ratio) events, E (emergency stop) events and O (delay multiplication overflow) events are all able to generate interrupts. These cases and others such as software Demagnetization and setting the carry bit after a computation are all managed by the software. They are described in the last part of this document.

5 SPECIAL FEATURES

5.1 R INTERRUPT (MTIM TIMER RATIO CHANGE)

There are two types of R event interrupts; R+ event and R- event. The type of interrupt depends on the value of the MTIM timer counter when the event is detected.

Note: For more information concerning R event interrupts, refer to Application Note 1082.

If a Z event is detected while the MTIM timer counter is between 55h and FFh, the event is processed normally.

If the timer reaches FFh before the Z or D event is detected, an R+ event (timer overflow) occurs. In this case, the clock needs to be slowed down, so the ratio is increased.

If a Z event is detected before the timer has reached the value 55h, an R- event (timer underflow) occurs. In this case, the clock needs to be accelerated, so the ratio is decreased.

Figure 17 and Figure 18 show what is done by the software when R+ or an R- event occurs.

As shown in Figure 11, when an R event is detected, the software checks the interrupt flags to determine if it is an R+ or an R- event.

If RMI flag is set, it is an R- event and the software goes to the Z_55 routine (Figure 17).

If RPI flag is set, it is an R+ event and the software goes to the Timer_ovf routine (Figure 18).

Figure 17. Z_55: Internal 8-bit Timer Underflow

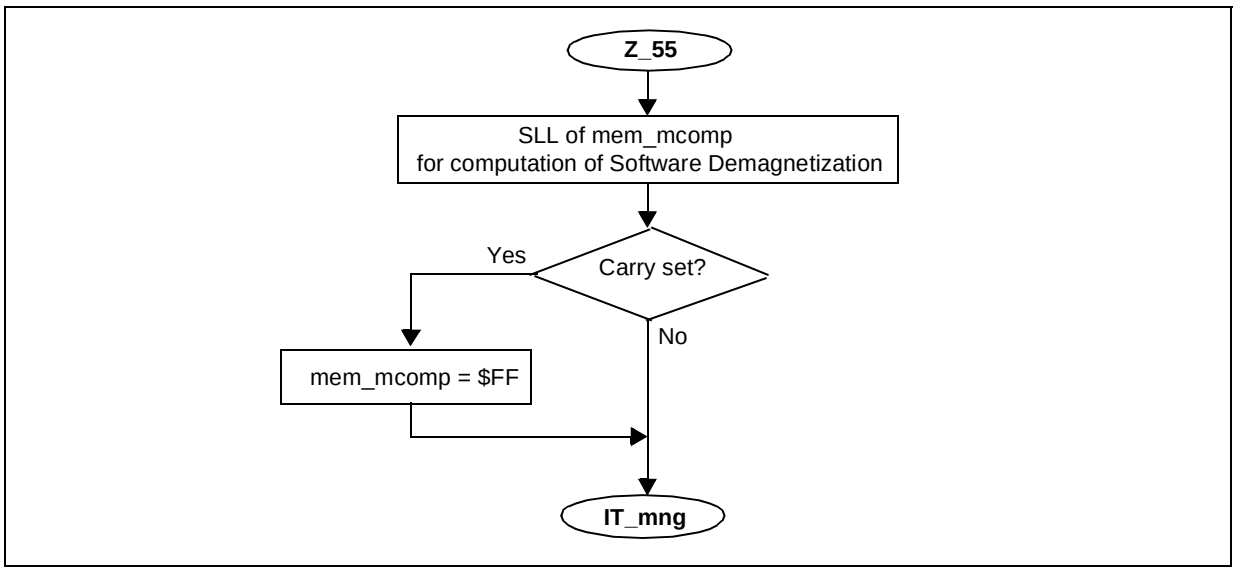


Figure 17 shows what takes place in the event of a timer underflow (an R- event). For all practical purposes, the timer ratio is automatically decreased by the microcontroller. Therefore, all the values are multiplied by two for the current step in order to avoid computation errors.

This means that the previous Zn and Z times are multiplied by 2, as well as the Demagnetization time value in the MDREG register.

If the carry is set during the computation, the maximum value is stored in the MCOMP register.

Then the software returns to the IT_mng routine (Figure 11), checks again for interrupt flags and returns to the main program and waits for another interrupt.

Figure 18 shows what is done when a timer overflow occurs (an R+ event). For all practical purposes, the timer ratio is automatically increased by the microcontroller. Therefore, all the values are divided by two for the current step in order to avoid computation errors.

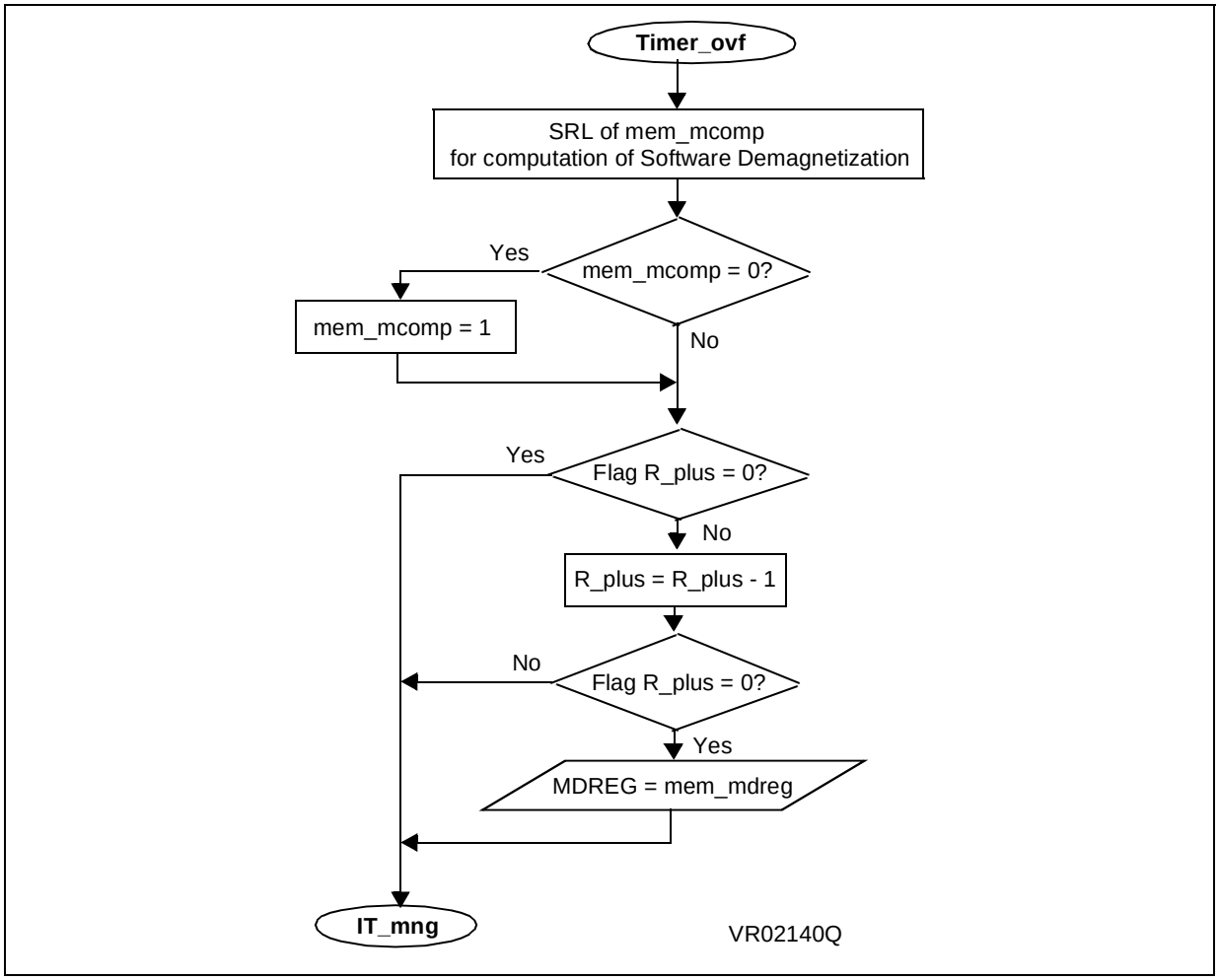
This means that the previous Zn and Z times are divided by 2, as well as the Demagnetization time value in the MDREG register.

The timer is reset to its middle value, i.e. 7Fh.

After division, if the value in the MCOMP register is equal to zero, the software stores a minimum value of 1 in the register.

Then the software checks again for the R+ event flag and returns to the IT_mng routine (Figure 11).

Figure 18. Timer_ovf: Internal 8-bit Timer Overflow



5.2 SOFTWARE DEMAGNETIZATION

In the software included with this application note, if Auto-switched mode is enabled, every second step will use software demagnetization. Note that this is a software example and that the fact that every second step uses software demagnetization can be changed.

Therefore, each step is checked by software for hardware or software demagnetization (set using the HDM or SDM bits in the MCRB register). This is shown in Figure 14.

Figure 19 and Figure 20 show what is done by the software in the event of software demagnetization.

Figure 19 shows the first part of the software demagnetization. In the event that a software demagnetization takes place in the current step (step n), a hardware demagnetization must have taken place in the previous step (step n-1). Therefore, Dn-1 was a hardware demagnetization.

The values in the MDREG (n-1) register for the hardware demagnetization and the MCOMP(n-1) register for the computed delay have been memorized.

The value in the MDREG (n-1) register represents the time between the previous Z event (Z n-2) and the End of Demagnetization event at step n-1 (Dn-1).

The real hardware demagnetization time is the time between the commutation event (n-1) and the End of Demagnetization event (n-1). So, the first thing done by the software is to compute the value of the real hardware demagnetization (HDM) event from the previous step:

HDM (n-1) = MDREG(n-1)-MCOMP(n-1)

Then the software checks if this value is greater than 0. If not, a minimum value of 1 is set. Then, to compute the software demagnetization time for the current step (step n), the hardware Demagnetization time is multiplied by a correction factor (chosen arbitrarily by the software programmer) to ensure that the software Demagnetization time for the current step will be greater than the hardware demagnetization time from the previous step (for safety purposes to prevent the End of Demagnetization event from occurring too soon):

SDM (software Demagnetization value) = HDM*1.25

Then the software checks if this value is greater than 255. If yes, this value is forced to 255.

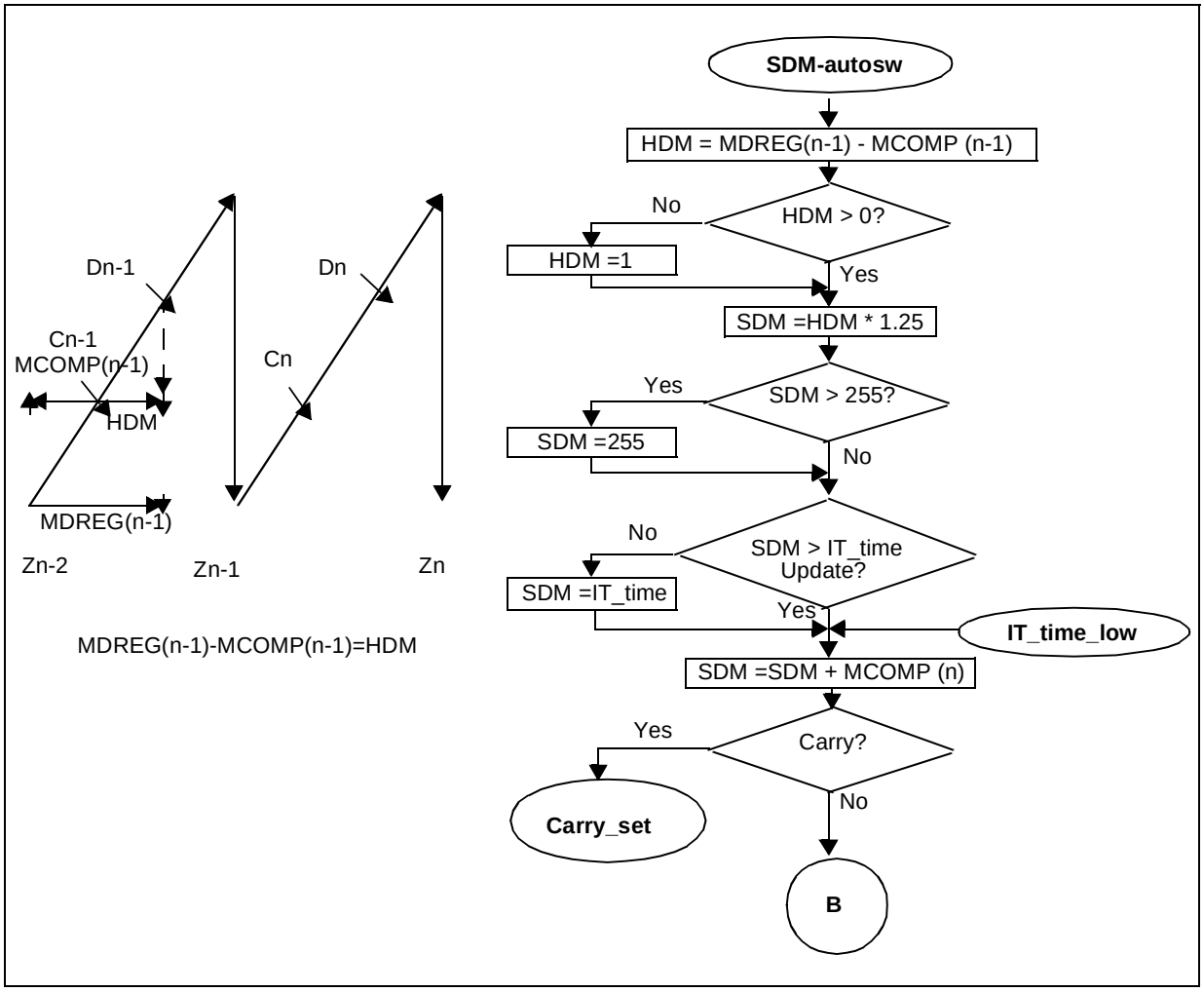
The software then checks if the value of the software Demagnetization time is greater than the time needed to process the interrupt routine (IT_time). If not, this interrupt processing time is set as a minimum value.

This is done because if the value in the internal 8-bit timer is greater than the value of the software Demagnetization when the routine is completed, the End of Demagnetization event will never occur since the value of the software Demagnetization would have been already reached by the internal timer before the end of the routine.

Then, the value of the delay between the Zn-1 event and the next commutation event Cn is added in order to obtain the time between the Zn-1 event and the software End of Demagnetization event.

If the carry has been set by these computations, the software goes to the carry-set routine (Figure 21). Otherwise, the software Demagnetization routine continues (Figure 20).

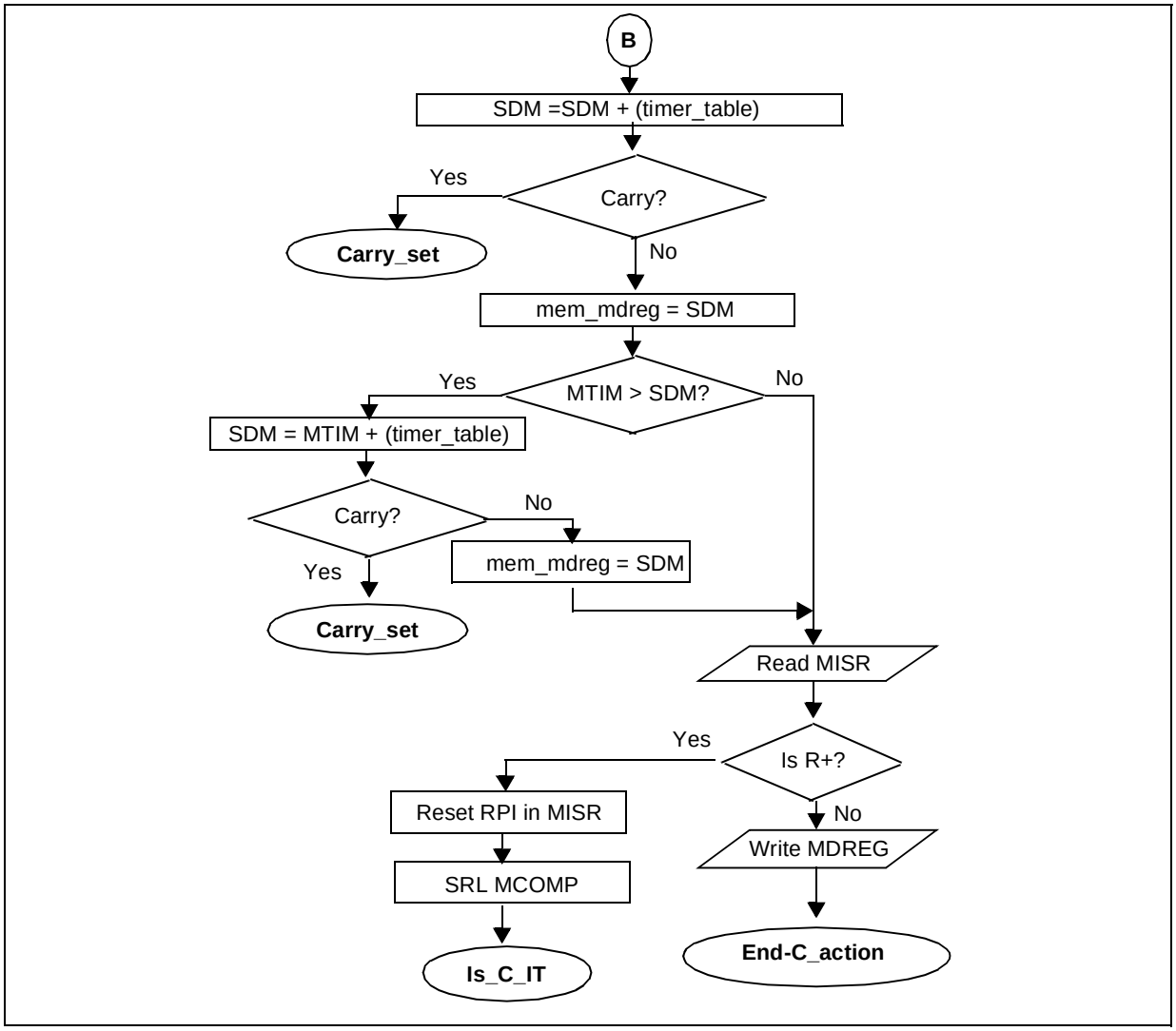
Figure 19. Software Demagnetization (First Part)



The software adds another correction factor to the software Demagnetization time corresponding to the time needed to load the register. If the carry has been set, the program goes to the carry_set routine. Otherwise, the actual value for the software End of Demagnetization event is memorized and the software checks if this value is greater than the current value of the MTIM timer.

If the value set for the software End of Demagnetization event is greater than the current value of the MTIM timer, a value greater than that of the MTIM timer is set for the software Demagnetization time.

Figure 20. Software Demagnetization (Second Part)

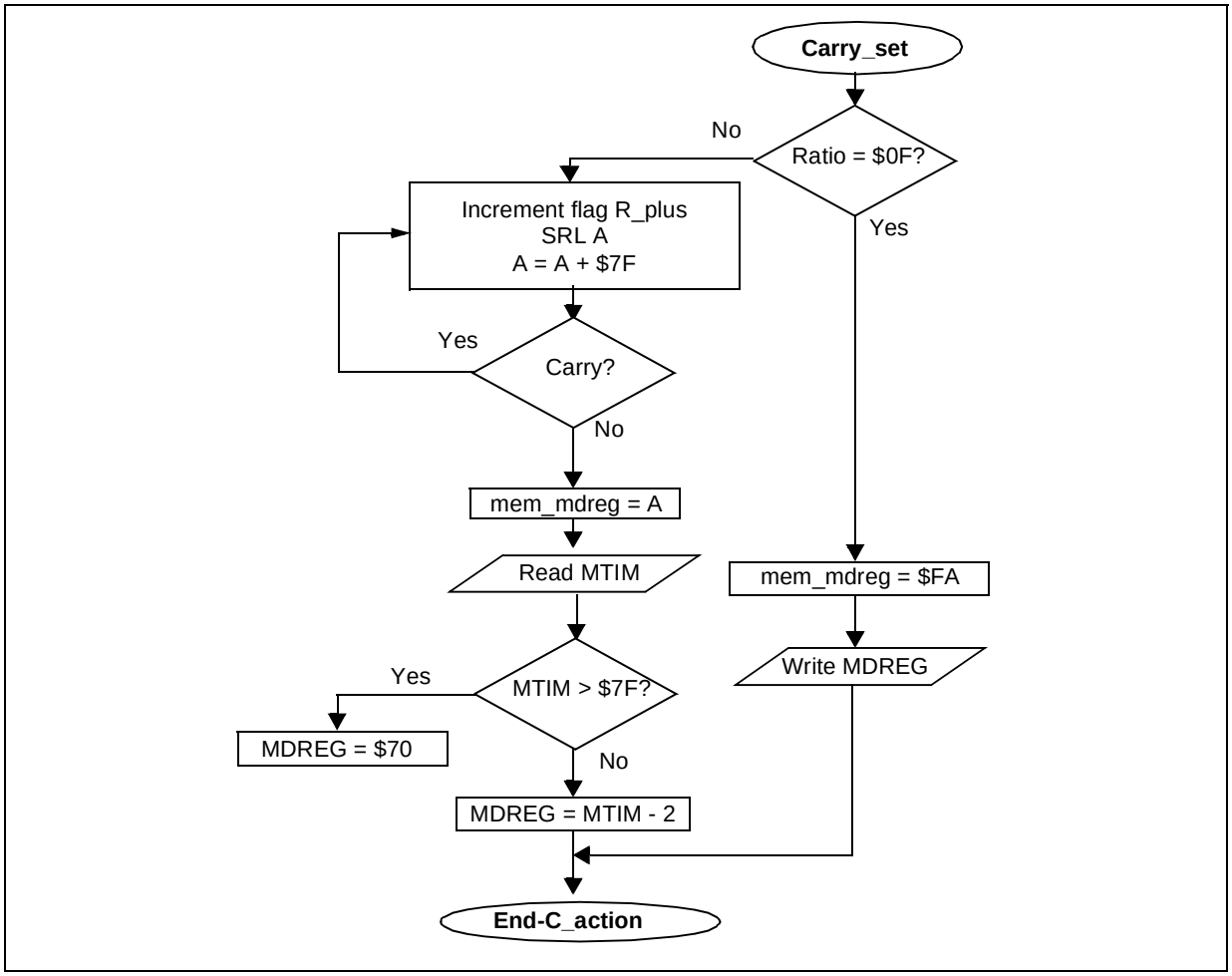


If the value set for the software End of Demagnetization event is less than the current value of the MTIM timer, the software checks if the flag for an R+ event is set.

If the flag for an R+ event is set, the flag is reset and the software returns to the beginning of the C interrupt routine. Otherwise, the value is written in MDREG register, then the software goes to the End_C_action routine to prepare the data for the next C event and then returns to the main program.

If the carry has been set during the computation for the software End of Demagnetization event, the software enters the carry_set routine as shown in Figure 21.

Figure 21. Carry Set Routine Flowchart



If the carry has been set, the software End of Demagnetization event will occur after the time the MTIM timer would have reached the value FFh. This will cause an overflow of the MTIM timer (an R+ event). Therefore, the software checks if the ratio of the MTIM timer is set to the maximum value.

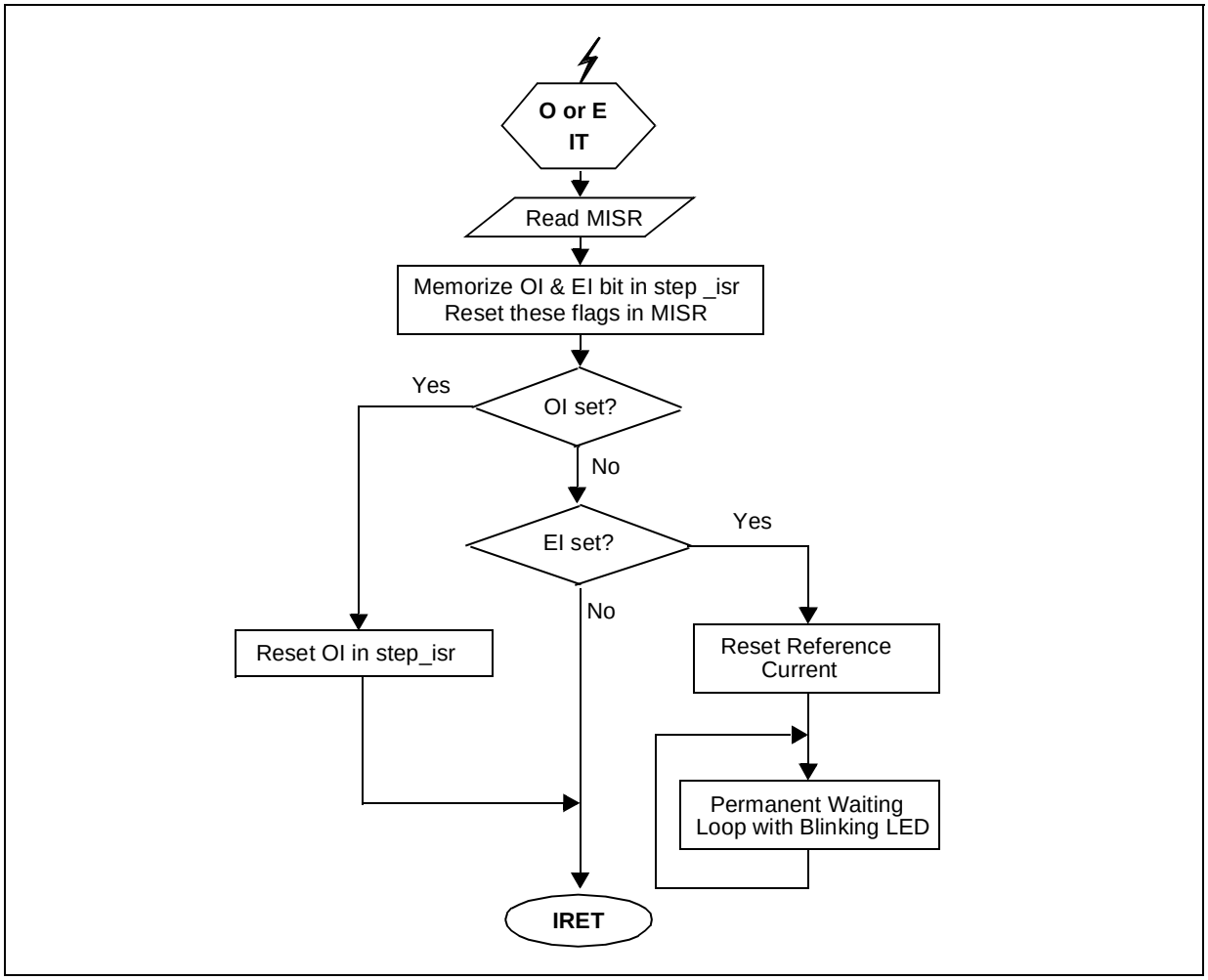
If the ratio of the MTIM timer is set to the maximum value, an arbitrary value is memorized for the software End of Demagnetization event and is written in the MDREG register.

If the ratio of the MTIM timer is not set to the maximum value, the R_plus flag is incremented, meaning that the overflow will occur during the software Demagnetization. To prevent this from occurring, the clock must be slowed down, i.e. the ratio must be incremented. The value of the ratio is updated to the new future ratio by dividing by 2 the value for the software Demagnetization. Then, when the ratio changes, the MTIM timer is reset to its middle value (7Fh), which is added to the software Demagnetization value. This new value is then memorized in the mem_mdreg variable and the MTIM timer value is read. To ensure that an End of Demagnetization event will not take place before the ratio change (when the R+ event will

occur), a value lower than the MTIM value is written in the MDREG register. The next interrupt will be an R+ event. At that time, the MDREG register is written.

5.3 O AND E INTERRUPT ROUTINES

Figure 22. Multiplier Overflow and Emergency Stop Interrupt Routine



If a multiplication overflow is detected when computing the delay between the zero crossing event and the next commutation, nothing is really done by the software. The interrupt flag is reset and the software returns to the main program.

For the emergency stop interrupt, the reference current is reset and the software enters a permanent loop that waits for the end of the emergency stop signal.

6 SUMMARY

Figure 23. Software Overview

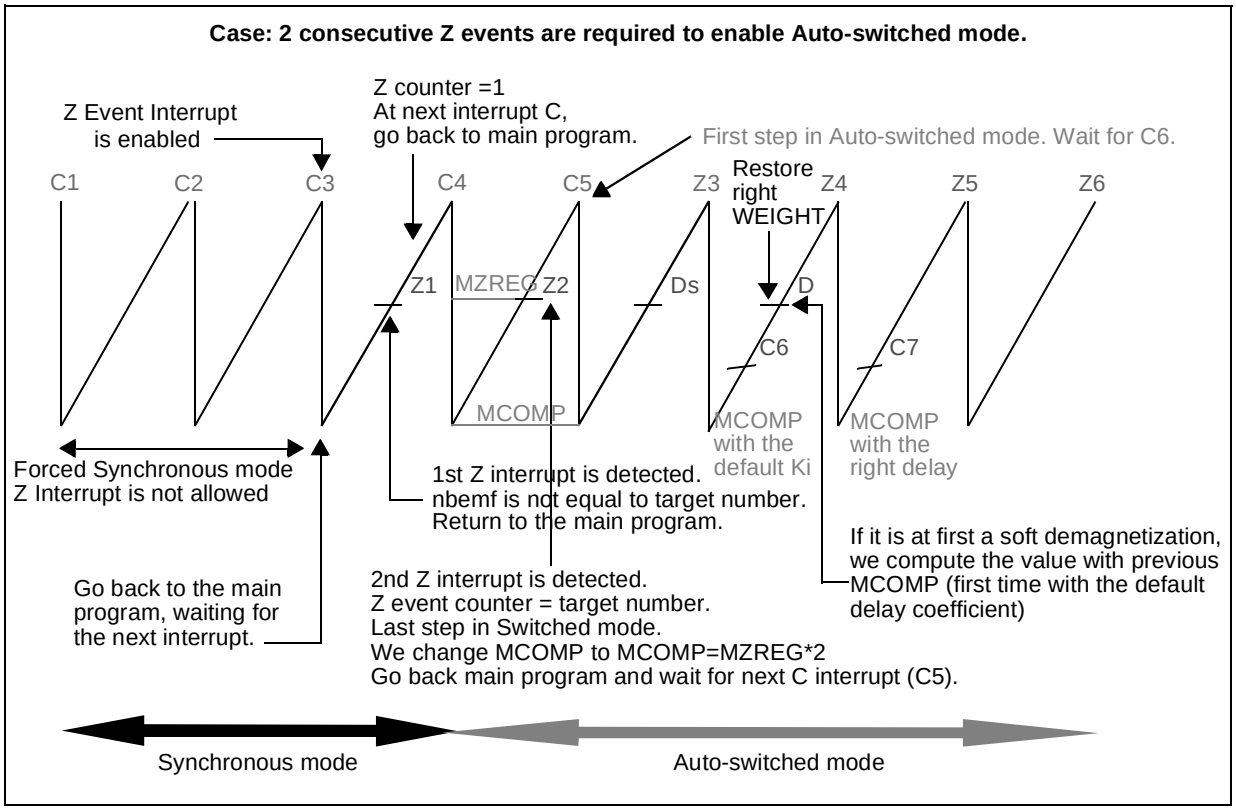


Figure 23 shows how the software behaves in the start-up sequence, when the motor is passing from Synchronous mode (switched mode) to Auto-switched mode. This figure illustrates the case where the software programmer wants 2 consecutive Z events before enabling Auto-switched mode and 3 steps in Forced Synchronous mode.

At C1, the motor is in Forced Synchronous mode. In this phase, the Z event interrupt is disabled. At the end of Forced Synchronous mode (set by software), the Z event interrupt is enabled (C3). The software then returns to the main program and counts the number of consecutive Z events detected.

When the second consecutive Z event is detected (Z2), the target number of consecutive Z events (set by software) is reached. Z2 is centered on the current step and the software memorizes that this is the last step in Synchronous mode.

At the following step, the motor enters Auto-switched mode and the software memorizes that it is the first step in this mode and the MTIM timer is now reset on the Z event and the D event interrupt is enabled.

Note that during all steps in Synchronous mode, the D event interrupt is not enabled and all End of Demagnetization events are software D events.