RL78/G1F Group

R30AN0249EU0100
Rev.1.00
Jun 24, 2016

Initial Position of Sensorless BLDC Motor

Introduction

When starting up a BLDC motor, it is required that you know the rotor angle relative to the stator windings in order to produce torque in the correct direction. This application note provides a method for determining the rotor position of a non-moving BLDC motor. It describes both the alignment technique and the method of "exciting" the winding with known patterns and determining the rotor position by reading the currents during the excitation period which provides both the location of the iron rotor as well as the direction of the flux, thus providing position within the 60° required to produce starting torque correctly

Target Device

RL78/G1F, 64 pin version (R5F11BLE)

Contents

1. Overview

This application note provides the theory and accompanying demo program for a brushless DC motor (BLDC) using the RL78/G1F microcontroller. The demo program will be described from an “Abstract level”, and the reader is referred to the C project files and the Doxygen folder within the project for code documentation. The app note will focus more on the theory of operation and the RL78/G1F features which allow the implementation.

In this application note, wherever possible, we will use actual scope shots (or DSO data) rather than simulated or mathematically created data.

NOTE: Although this Application note was verified with a block wound motor it should operate with a sinusoidal wound PMSM motor.

Operation checking device

Operations of the sample programs have been checked by using the following device.

- RL78G1F (R5F11BLE)

Target sample programs

The target sample programs of this application note are as follows.

(1) RL78G1F_INITIAL_POSITION for RL78G1F (R5F11BE) for T2001 Inverter

(Complementary PWM Positive Side, GPIO Negative Side, see section 0 for details)

NOTE: This demo is used to show initial position determination, however, we clearly need to demonstrate the motor running from that position. The algorithm here is based on the CMP BEMF application note. See section 8 References, item 2

1.1 Development environment

Table 1-1 shows development environment of the sample programs explained in this application note.

Table 1-1 Development Environment of the Sample Programs

	Sample software	MCU	Inverter board	Motor	E2studio version	Tool Chain
Low-voltage version	(1)	R5F11BLE	T2001 Note 1	Portescap[2]	V4.1.0.018	IAR for RL78 V1.40.6.937

For purchase and technical support of the low-voltage inverter board T2001, contact Sales representatives and dealers of Renesas Electronics Corporation.

Notes:

1. T2001 is a product of Desk Top Laboratories Inc.
2. Portescap Size 23 BDLC Motor, C-230012-20A
3. Some testing was done on an “off-the-shelf” BLDC power tool motor also.

2. System overview

Overview of this system is explained below.

2.1 Hardware configuration

The hardware configuration is shown below.

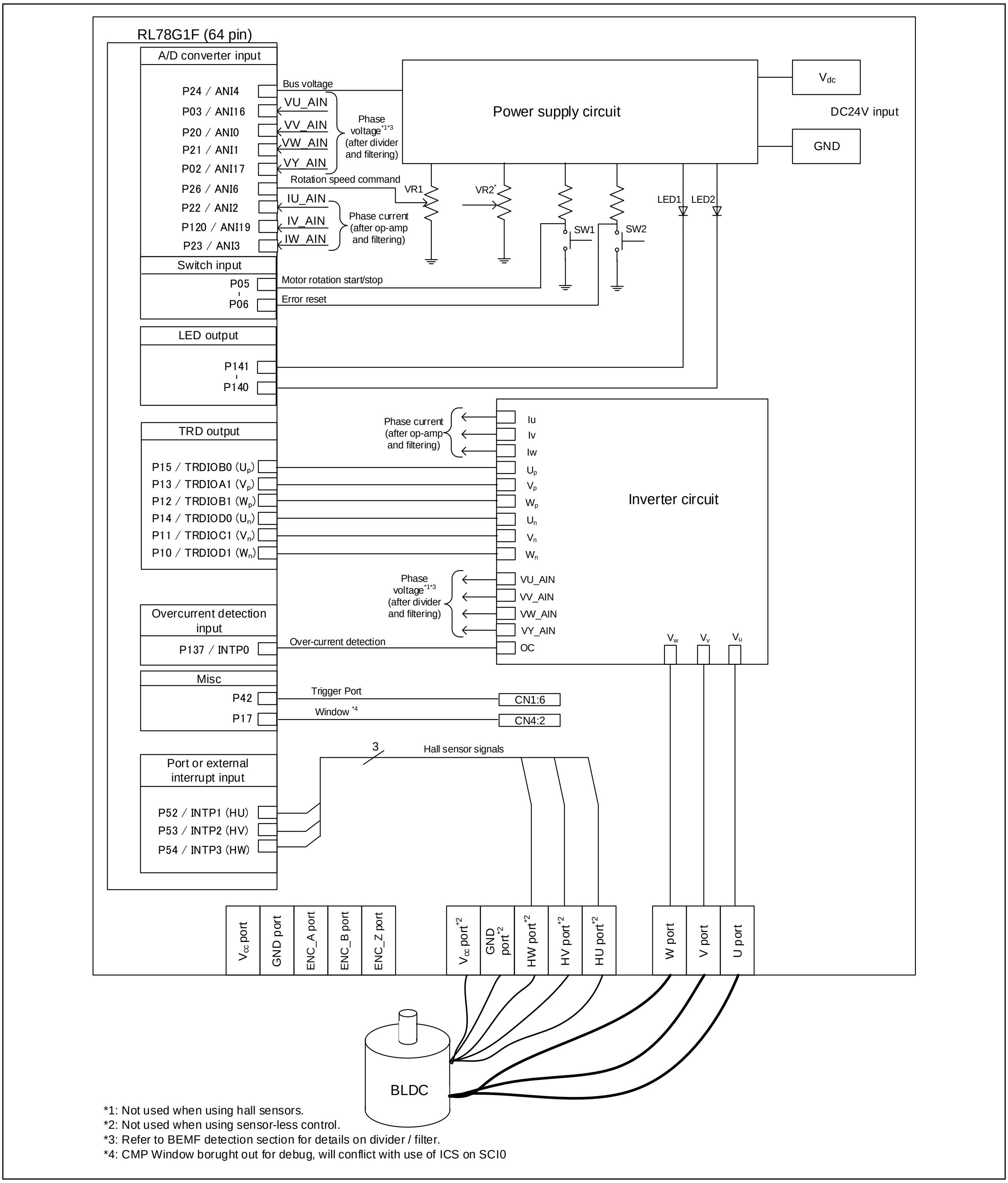

Figure 1: Hardware Configuration Diagram

2.1.1 Alignment Hardware Requirements

The only hardware required to do the alignment is the inverter drive itself. We will be using the standard complementary drive on the positive side of the motor and GPIO on the negative side of the motor. Alignment by its nature is an "open loop" scheme for forcing the rotor position. Of course the Over-current hardware will still be in operation in the event the software inadvertently over-drives the motor during the alignment phase.

2.1.2 Current Mode Initial Position Hardware Requirements

In addition to the inverter drive section of the hardware, this mode of initial position determination requires 3 shunts and op-amps to read the current through the windings of the Motor (see Figure 2, one motor leg shown). The details are not shown, but the shunt and op-amp gains will be determined by the maximum running current of the motor being used. Since the amount of current you can drive is based on the motor and inertia of the system, it is typically not a problem with reading the currents used by this algorithm using the 10 bit ADC provided in the Rl78/G1F device.

DESIGN NOTE: Since the shunt / op-amp combination has no hardware filtering (i.e. no integrating capacitors), and the current through the shunt is only valid while the low-side MOSFET (not shown) is on, the conversion time and the motor time constant must be taken into consideration when reading all 3 shunts. Since the low-side is on for the entire duty cycle, we only need worry about the conversion time and motor time constant. Although the motor current cannot die off instantly, it will start to decay down based on the motor time constant (the slope will be very low). In the case of the RL78/G1F, the minimum conversion time at 5.0VDC is given as 2.125µs, so the conversion can be completed in about 4.2µs total. This time can be adjusted by adding some filtering to the shunt circuits, but at the expense of some phase lag in the readings.

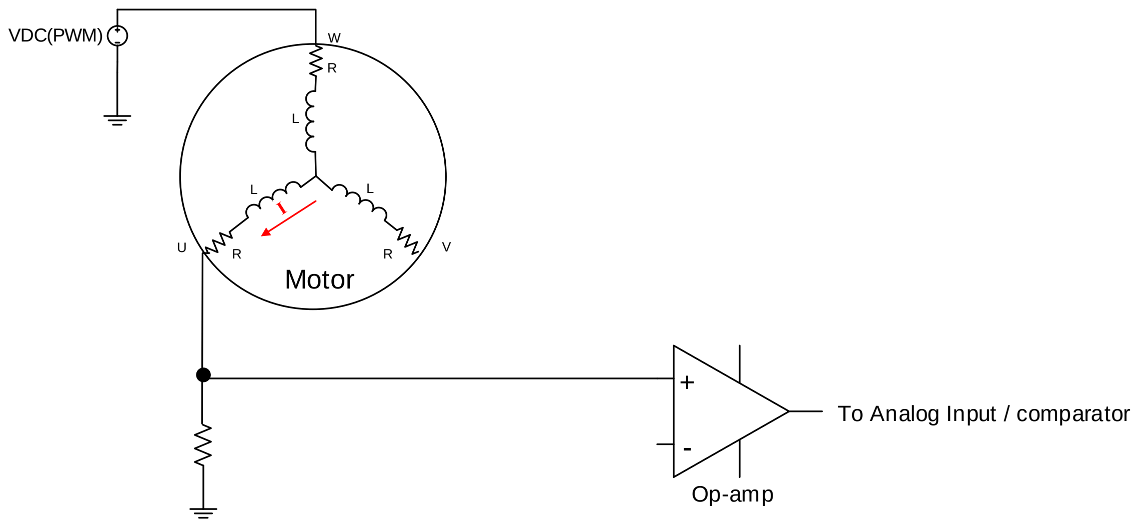

Figure 2: Motor current hardware model

2.2 Modulation Method

Figure 3 shows the modulation method used in this application. In this case positive side modulated, negative side GPIO. This is compatible with the algorithm mentioned in section 8, item 2 so we will not have to re-program the inverter timer when we switch to open loop mode of spinning the motor.

This modulation method provides:

- low-side drive always on during the step, so sampling "shunt" current is a little bit simpler.

Figure 3 shows the 6 STEPs of modulation, you can see the PWM modulating the positive side of the motor.

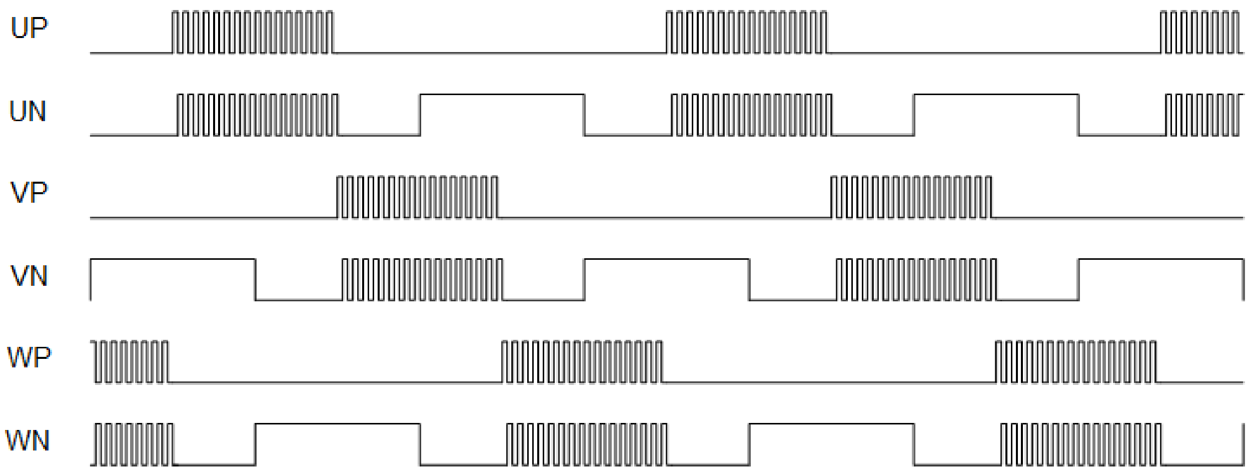

Figure 3: Modulation Diagram (active high drive)

2.3 TimerRD TAU trigger Multiplexing

Figure 4 shows the multiplexing provided on the RL78/G1F on TAU channel 0 to allow the various PWM signals of TimerRD to be the trigger. So this allows TAU channel 2 to be the Master of a Master/Slave pair of a "one-shot".

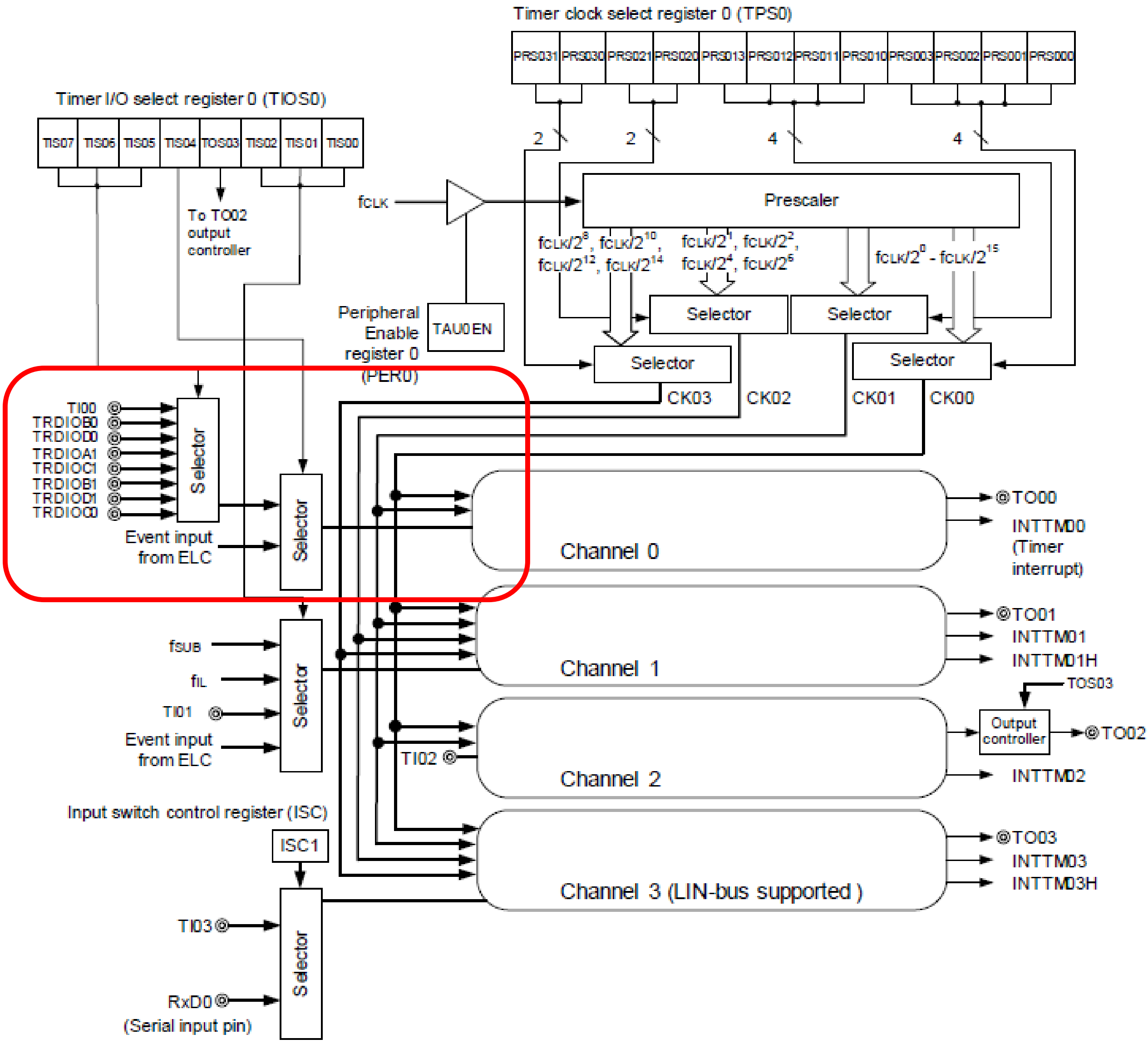

Figure 4: TAU / TimerRD signal multiplexing

The CMP Window we generate can be routed through the ELC and be used for other things. For example, we may use it to do the first trigger of the ADC for Vu voltage and then use the ADC interrupt to complete the conversions of the other required signals. For details on the use of this hardware "window", please see section 8, item 2.

3. Controlling Program Description

In this section we will give an abstract view of the motor control program. For details on the program the reader is referred to the Doxygen output folder (index.html) for code structure, variables, definitions, and call/caller details.

3.1 System States

Figure 5 shows the system states as defined by a call to the Motor Event Function (`R_MTR_ExecEvent()`) and the events it can be called with.

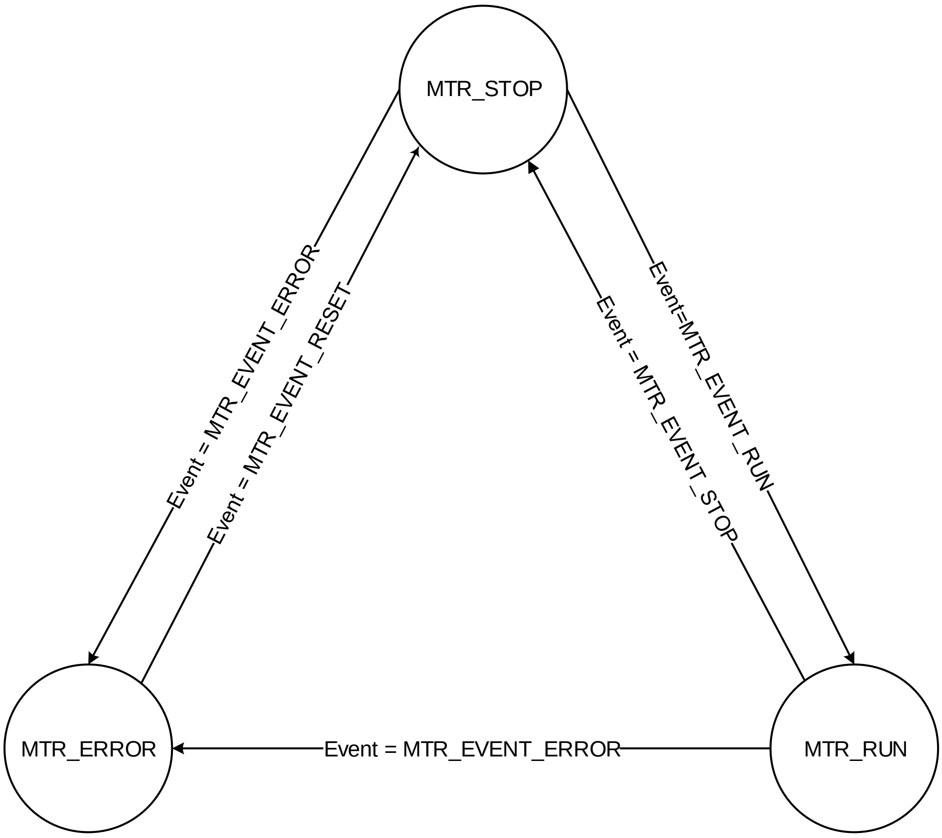

Figure 5: Motor Event States

A call to the function R_MTR_ExecEvent() also makes an indirect (function table) call to the default actions required by the transition into the new state.

For example, a MTR_EVENT_ERROR could be any number of items that indicate the motor or the system is not in a condition to run properly, including but not limited to Over-voltage on the VBus, Over-current while motor is running, and Over-speed.

NOTE: For the purposes of the initial position demo, we will only discuss the actions of the INTIAL POSITION functions in terms of being in the MTR_RUN state. For details on the full Motor Algorithm, please see section 8, item 2.

3.2 Motor States

Figure 6 shows the motor states for this algorithm. The motor state transition are caused by various values and events within the system as the motor runs, stalls, etc.

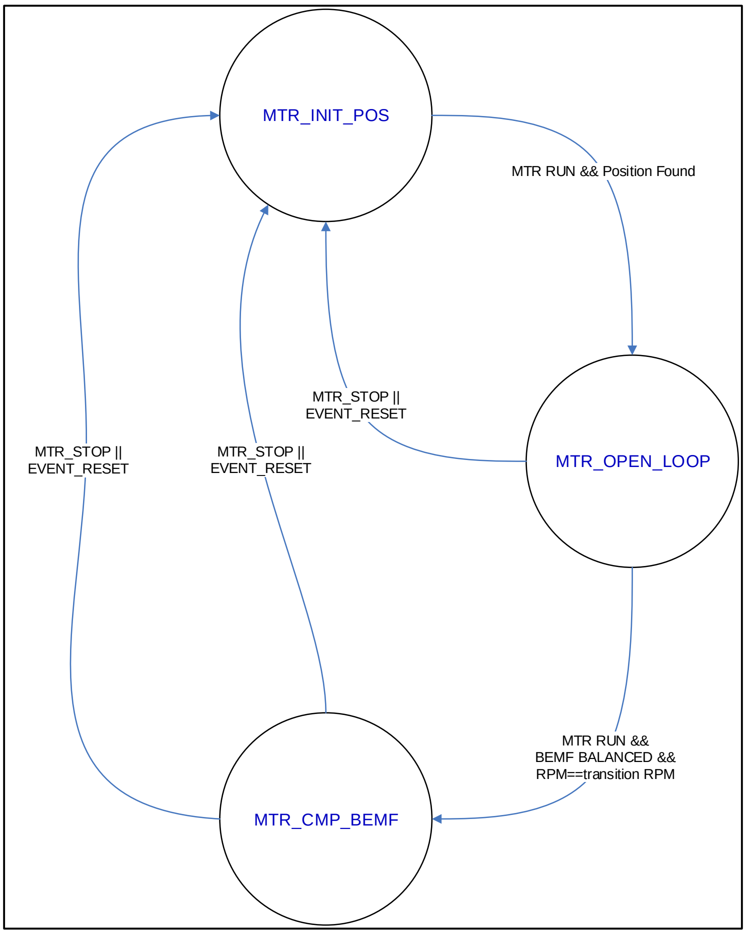

Figure 6: Motor State Diagram

Motor State transitions are a little harder to track, so a brief description of each.

MTR_INIT_POS: This is the starting state for the Motor operation. It is here after a MTR_EVENT_RESET. It remains here until the System State goes to the MTR_RUN mode at which time this state determines the initial position of the rotor. In the case of this Demo, the method used is chosen by a define for POSITION_MODE in the mtr_common.h file.

```
// Set this to one of the above values
#define POSITION_MODE USE_ALIGNMENT
```

where the values can be `USE_ALIGNMENT` *or* `USE_CURRENT`

Once the initial position of the rotor is determined without an error, it transitions to the OPEN Loop state.

MTR_OPEN_LOOP: In this state, the motor starts a pre-determined profile to spin the motor to a given RPM. For detail on transition to the CMP_BEMF mode, see the application note section 8, item 2.

MTR_CMP_BEMF: The motor stays here until a MTR_STOP event or a MTR_EVENT RESET. Stop is usually done by a commanded RPM of 0 or SW1 turned off. EVENT_RESET can be caused by any number of dynamic events, but most common is over-current.

IMPORTANT: For the purposes of this INITIAL POSITION demo program we will only be concerned with the MTR_INIT_POS state and the transition from it to MTR_OPEN_LOOP. For actual motor operation within the algorithm, the reader is refer to section 8, item 2.

3.3 Initial Position State machines

3.3.1 Alignment State Description

Once the motor is running in either OPEN loop of Comparator mode, the commutation is a finite machine of 6 states, basically either an up or down counter depending on the direction of the motor. The state machine is given in Figure 7. The length of time in the Alignment state is empirically derived for the motor and the inertia of the load. It is set in software by two definitions.

```
#define PATTERN_WAIT_CNTS   400
#define PATTERN_TOTAL_CNTS  10
```

where

- `PATTERN_WAIT_CNTS` *is the number of times in the current pattern and*
- `PATTERN_TOTAL_CNTS` *is the number of total patterns.*

Since we do a two step alignment, in the case of 10, we will drive 5 of one pattern and 5 of the other for 400 carrier counts each to let current build.

We align so that STEP 5 is the next state, so the current state will depend on the direction the motor will be commanded to go.

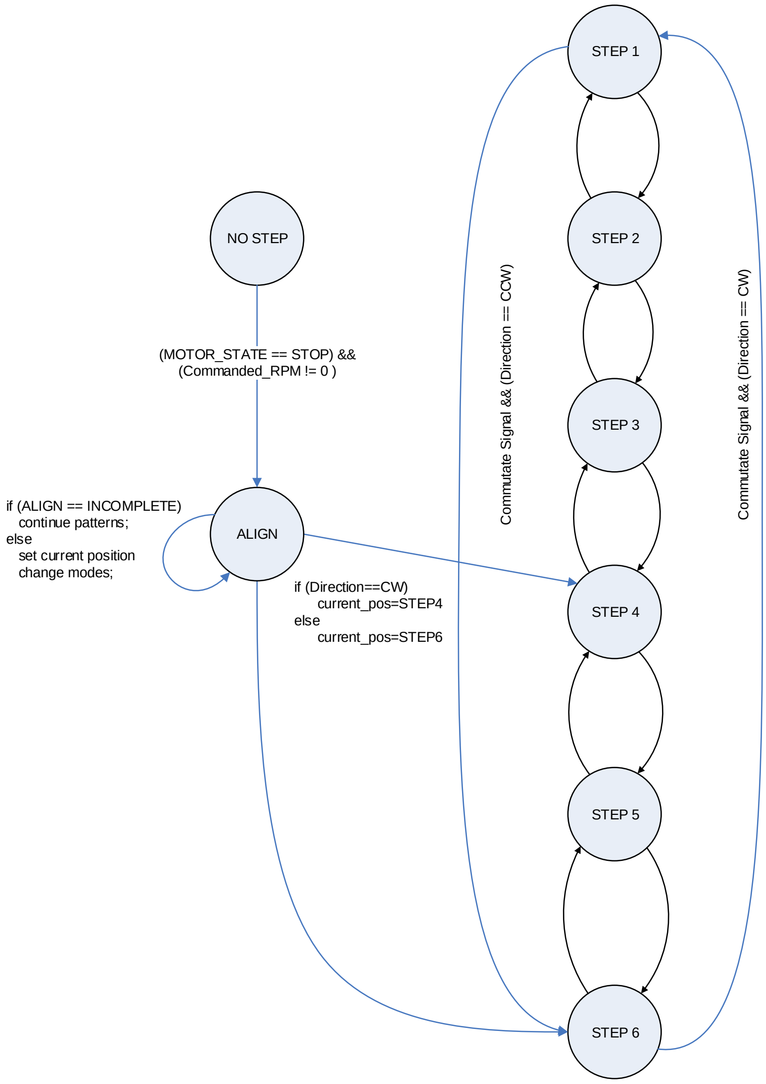

Figure 7: Motor 6-Step States Diagram, Align

3.3.2 Current Mode State Description

Once the motor is running in either OPEN loop of Comparator mode, the commutation is a finite machine of 6 states, basically either an up or down counter depending on the direction of the motor. The state machine is given in Figure 8. It is set in software by three definitions.

```
#define    PATTERN_WAIT_CNTS   5
#define    NUM_POS_PATTERNS    12
#define    PATTERN_TOTAL_CNTS  12
```

where

- `PATTERN_WAIT_CNTS` *is the number of times in a single pattern (current building)*
- `NUM_POS_PATTERNS` *is the number of total patterns. 6 are excitation patterns and 6 are dead-time patterns to let the current die out (motor time constant related) and in case of adjacent patterns (could cause "shoot-through" currents).*
- `PATTERN_TOTAL_CNTS` *is the number of total patterns before we should have a decision.*

The length of time in the current mode state is basically the `PATTERN_TOTAL_CNTS` of the definitions below times the carrier period. So for the example above:

*12 * 50μs = 600μs*

The main difference in the state machine for this method and the state machine for the ALIGN method is that we determine the position rather than force it. So the open loop commutation can jump in at any STEP state.

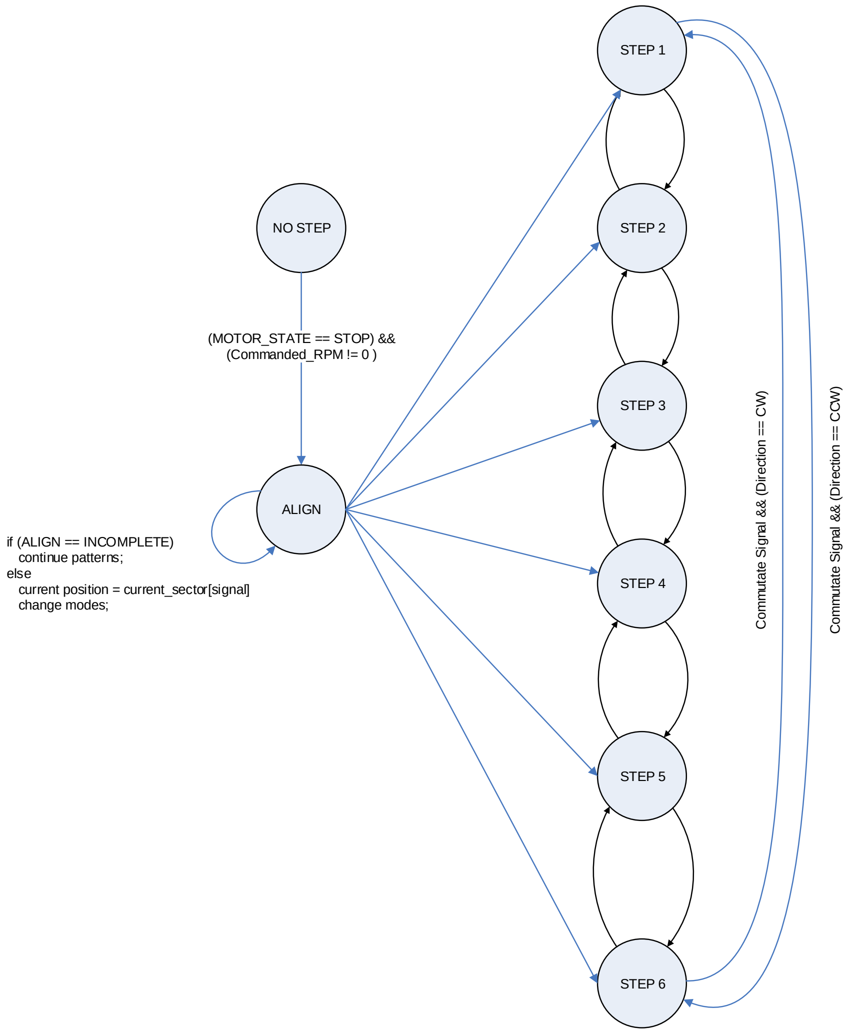

Figure 8: Motor 6-Step States Diagram, Current Mode

4. Alignment Mode

4.1 Method Description

The alignment mode is one of the simplest methods for putting the Rotor into a known angle relative to the stator windings. This is done by exciting the winding to produce torque in a known direction.

In most cases you can excite just one pair of winding and the motor will move into a given position. The problem being, that in certain cases the excitation pattern you use will produce zero torque as the rotor is at exactly 180 degrees to the stator. For this demonstration, we will use the two excitation pattern method to move the rotor into position so we will always be in a position to produce torque.

Along with producing torque to move the motor, it must be ramped in such a way as to move the motor slowly into position and not over-shoot the sector desired (i.e. it must stop / be stable at the given position to start the motor properly in OPEN LOOP start-up).

We will apply two patterns, each of which will ramp the duty cycle based on a look up table, divided in half for the first pattern and then the second pattern.

```
int16_t align_duty[PATTERN_TOTAL_CNTS]
```

Each pattern will typically ramp at the same rate.

The patterns being driven for this demo are:

- Up, Vn
- Wp, Un

Figure 9 shows both the two pattern excitation current and the slow ramp being applied to move the rotor into a “known” position.

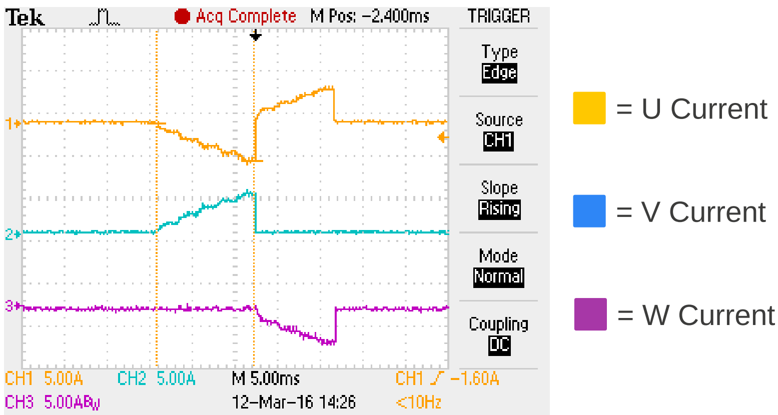

Figure 9: Current for Alignment patterns

This will allow the motor to turn and “align” with the rotor. The drive values are empirically derived based on the inertial load and the motor drive capability. It is important to allow the motor to “creep” into position and lock so that a known position is established. If the motor is driven too hard, the motor will overshoot the position and the ability to start will be lost.

The alignment method is great for applications where the slight motion of the rotor is inconsequential to the System Design. Clearly may not applicable to Traction Motors, Power Tools, etc. Great for things like ventilation fans, blowers, some pumps, etc.

PROS:

- Simple to implement in software (high performance MCU not required)

CONS:

- Times and current empirically derived for a given range of start-up inertia
- Rotor will move (direction unknown) during start-up

5. Current Mode

5.1 Flux influence on Current

Before we can describe our method, we need to see a simple picture of how the flux direction affects the current. If we look at the total flux in the motor (see Figure 10), it will be a combination of the Permanent Magnet (PM) flux and the flux created by exciting the stator, essentially the summation. So for a given current the flux will rise to a certain point. If we excite the stator for a specific value, the current that creates flux in the same direction will get there faster (faster rise time). The converse of this is that if we excite the stator with a specific voltage for a specific time, but in two directions, the current producing flux in the same direction as the PM flux will rise to a higher value in that given time. This is shown pictorially in Figure 11.

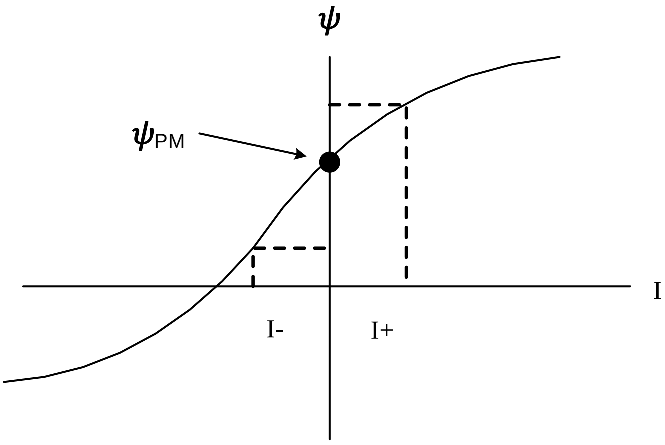

Figure 10: Total Motor Flux (ψ)

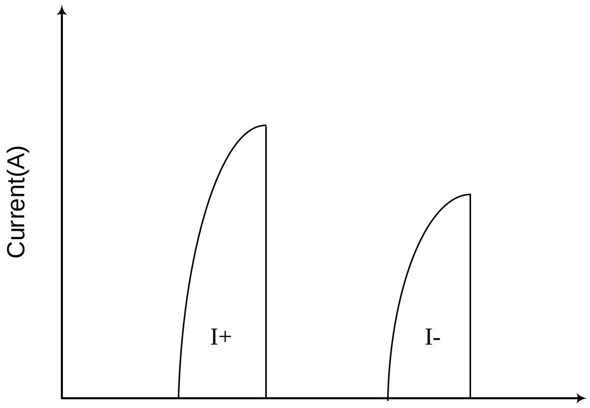

Figure 11: Current resulting from fixed time / voltage

5.2 Extrapolating to 3 phases

We know what the flux in the PM of the rotor will do to the current if we excite the stator winding in two directions. We do however have 3 phase (3 coils) to contend with. One problem we have is that we are using low-side shunt, so we can only measure the current on the winding where the low-side MOSFET is turned on. In order to determine the direction of flux in a particular winding, we need to drive it in both directions. If you look at Figure 12, you can see that when we drive phase B and C positive, we can measure phase current A directly (low-side shunt). However, if you look at Figure 13, it becomes obvious that we cannot measure A directly any longer and must take the sum of the currents in Phases B and C (Kirchhoff's Law). This means we need to drive 2 patterns to measure each winding for a total of 6 active patterns. We drive an inactive pattern between active patterns to allow the motor current to decay back to 0. By exciting all the patterns, it allows us to calculate the fields from the resulting currents (inline or opposing the PM flux as described previously), we should be able to discern the position of the rotor.

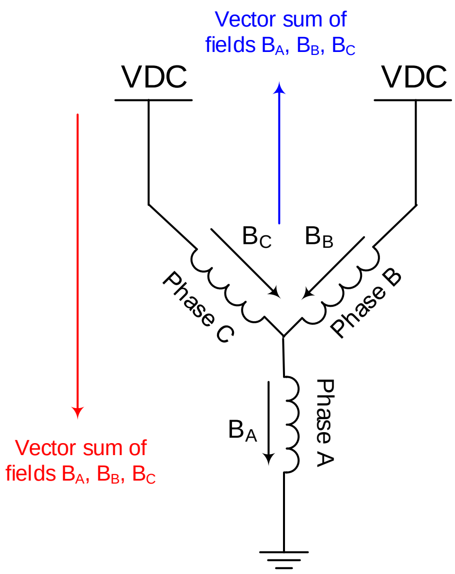

Figure 12: Measuring A current (directly) from Positive excitation of B and C

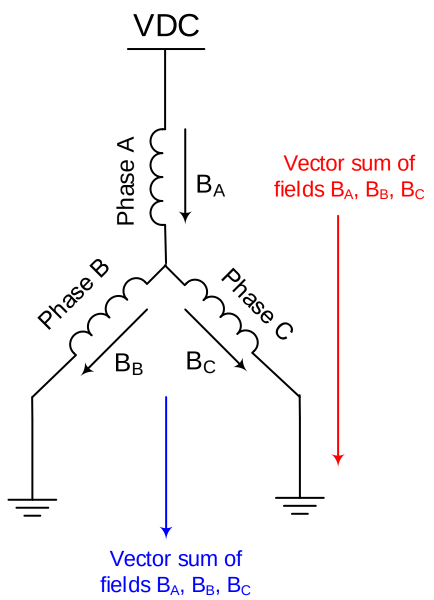

Figure 13: Measuring A current (indirectly) from Positive Excitation of A

5.3 Method Description

The current mode method is great for applications where any rotor movement in the wrong direction is intolerable to the System Design. May be applicable to Traction Motors, Power Tools, etc. Some pumps which cannot have the vane turn backwards may also benefit from this method.

PROS:

- Motor does not move using this method of excitation

CONS:

- A little more difficult to implement in software, but can still be done by a lower performance MCU.

In order to discern the rotor position, we will use the effects of the PM on the stator flux (and thus the current). We will need to excite the various phase in combinations that will result in calculating the rotor position relative to Phase A, B, and C. Again looking at , if we excite Phase A in both the + and negative direction it takes 2 patterns, doing this for Phase B and Phase C results in a total of 6 patterns.

We will apply the excitation for 10µs out 50µs (about 20% duty cycle, but will vary with the motor). This will easily allow us to read the current. The 6 patterns and the subsequent measurement, for exciting the motor are:

1) Up, Vn, Wn measure U_{PEAK} current (read V and W currents, calculate U_{PEAK})
2) Un, Vp, Wp measure U_{PEAK} current (read U_{PEAK} directly)
3) Vp, Un, Wn measure V_{PEAK} current (read U and W currents, calculate V_{PEAK})
4) Vn, Up, Wp measure V_{PEAK} current (read V_{PEAK} directly)
5) Wp, Un, Vn measure W_{PEAK} current (read U and V currents, calculate W_{PEAK})
6) Wn, Up, Vp measure W_{PEAK} current (read W_{PEAK} directly)

Figure 14 shows the currents for the 6 excitation patterns listed above. You can clearly see that there is a "unique pattern" for the currents showing a varying peak value for current as affected by rotor position, as well as "Dead-time" between excitations to let motor decay back to 0.

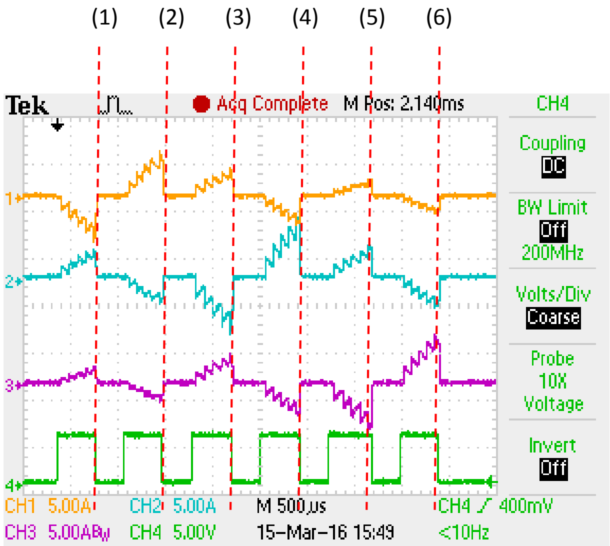

Figure 14: Motor currents for Current Mode patterns

NOTE: the excitation is only 6 patterns, but between patterns we need to allow the stator to return to "known state", so between excitations, we shut down the inverter. This actually results in 12 patterns, 6 of which are "dead-time" between excitations.

During these 6 excitation patterns we log the currents. Once the pattern generation is complete, we scan through the values and determine the axis where the current is MAXIMUM. Once we find the axis where the current is MAXIMUM, we can then determine the flux direction in that axis by comparing the plus (+) and minus (-) current flow values in that axis. These two determinations result in us knowing the 60° sector where in which the rotor lies. This is diagrammed in the flow chart in Figure 15. We can then enter the OPEN LOOP mode at the correct sector.

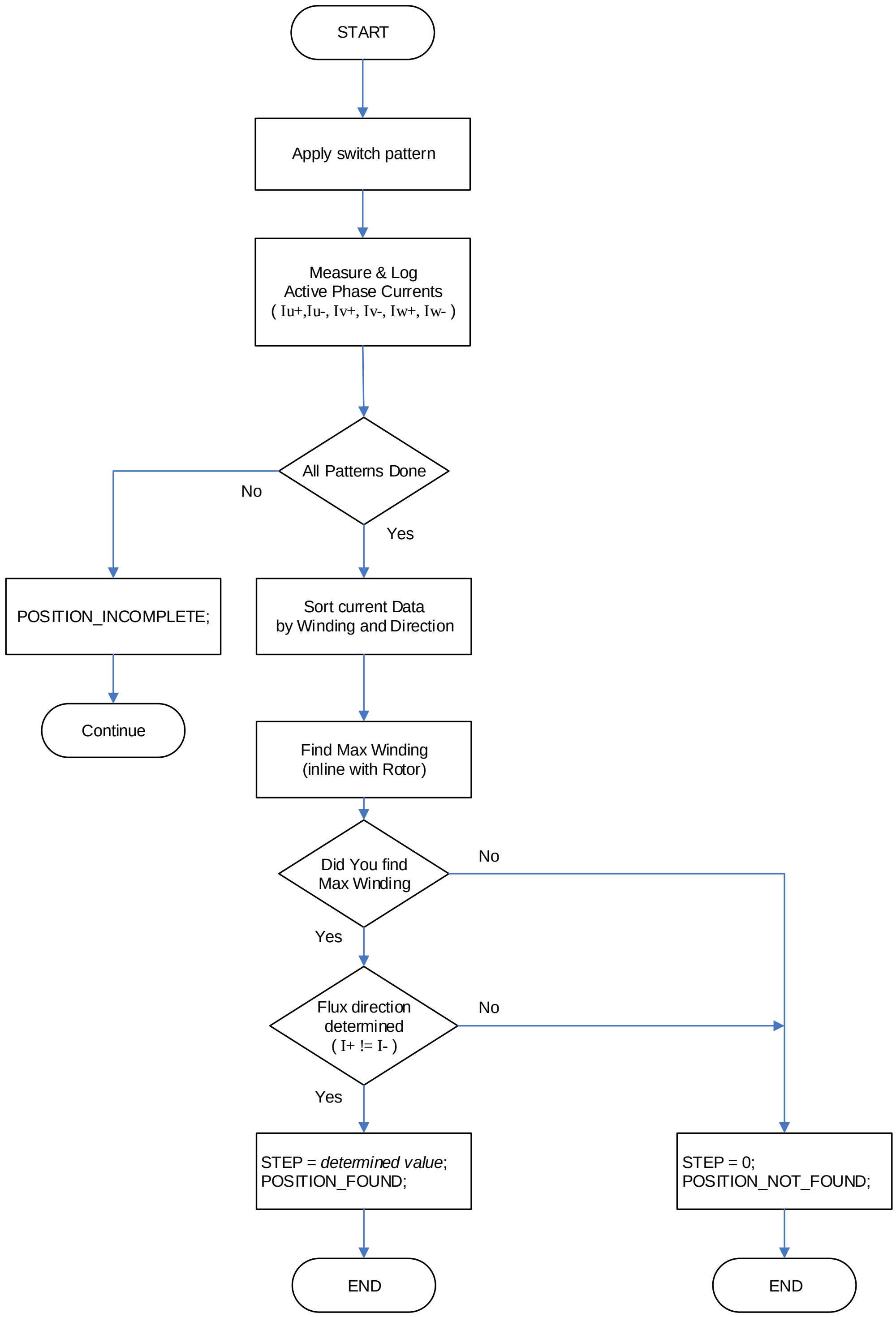

Figure 15: Flow Chart, Initial Position determination

5.4 Choosing Excitation Current

Depending on the motor, this may or may not be difficult, it largely depends on the R and the L of the motor. You goal should be to drive it high enough to make the difference in the axis currents observable (as in Figure 14, you can even see this on the scope), but not to drive the motor into saturation nor trip your over-current protection in your inverter. Due to the excitation pattern, you may hear a large “click”, but the motor should not move during this period regardless of the current.

5.5 Multiple position Scans

For motors with low R/L there may be trouble detecting the small current deltas between patterns. Remember, sometimes the deltas will be small and hard to detect given the ADC accuracy and noise in the system. If you have exhausted the current drive values and times (should be close to but not saturating the rotor core) and you have validated your sampling method (see section 9.1 Current sampling Verification) it may help make the make the Initial Position detection more robust by running multiple scans (i.e. some number of times) and count / bin the results, then use the STEP value with the largest number of HITs (a sort of voting method).

6. Miscellaneous Operations related to Motor Control

6.1 Over-current (INTP0 operation for this app note)

The demo is currently set-up to use the INTP0 signal as the overcurrent indicator. This is useful in the event you over-drive during alignment or Current Excitation for finding initial position. The platform that this was tested on was multi-shunt with the proper “window” comparator built onto the inverter. See reference 8, item 6

7. Demo Project

7.1 Importing and Building

1) Open the e2studio workspace you want project located in
2) Use the standard Import Feature of e2studio to Import an Existing Project
3) Browse (root directory) and select the sub-directory *Workspace\RL78G1F_IP_DEMO* where you unzipped the file.
4) Select *RL78G1F_ IP_DEMO* and FINISH.
5) If asked, Browse to your IAR install location and click APPLY, then OK
6) Project should look like Figure 16:

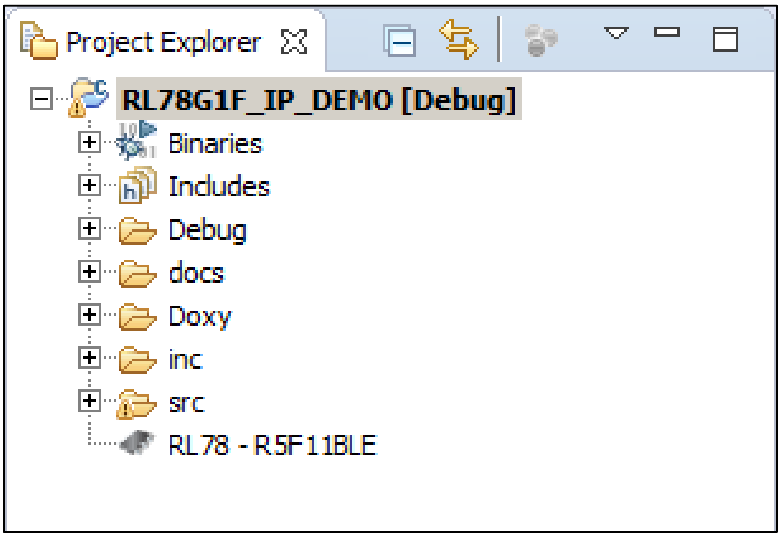

Figure 16: Project Explorer Panel after Import

7) Select menu Project Build ➔All (or ).

   NOTE: there may be some warnings due to the use of volatiles in the program.

7.2 Motor Selection

The project is built with parameters derived for the Portescap Motor. We have provided a means of switching to other motors. USER_MOTOR can be used when tuning for your particular motor, just change the define in mtr_common.h and search on USER_MOTOR for the areas that may need to be tuned / changed.

```
// Motor Options
#define USER_MOTOR 0
#define PORTESCAP 1
#define OTS_PT_MOTOR 2
// set the motor used here
#define MOTOR_USED  PORTESCAP
```

7.3 Tips and Tricks

7.3.1 Testing

Testing and validating the operation of initial position methods is a little tricky. Many motors will actually “catch” when started “open loop”. However, if you are trying to start without going in the wrong direction first, it is important to validate you initial position algorithm. Some method for validating:

- Log start-up of motor on Dyno. May be difficult, even if motor appears to start correctly you may just be getting lucky. If possible, you can align dyno with rotor when coupling, to get better results.
- Use identical motor but with HALL sensors *only for validation*. Position will be known within 60 which is the same as these methods shown in the demo (method used in testing this application note).
- Identical motor with Encoder aligned to the rotor (0° aligned to Index pulse and 360° mechanical position is known).

7.3.2 Warning on Motor Types

This algorithm is tested with a low-power motor and a high-power motor. One of the things they have in common is that they are not extremely low in inductance or resistance. The algorithm works by driving the motor close to saturation, but with a cycle time that will not overcome the inertia of the system (so the motor does not move). For motor with extremely low R-L characteristics, the concept still applies, however, the solution may be limited by the speed and resolution of the ADC and the number of S/H (i.e. ability to sample the two low-side active inverter legs simultaneously).

7.3.3 Expressions

Most of the State Machines use enumerated values for their state variables, so it makes it easy to “watch” these in the expressions window to determine the whole state of your “machine”. For example, in Figure 17, from our Auto-refresh variable (large R indicator) we can see that:

- Our system mode is in MTR_MODE_RUN,
- NO errors have occurred
- Our motor run mode is Initial Position Mode
- The temporary status of the position search is FOUND
- We are in STEP 6.

It should be noted, that the real time refresh rate is typically in the 100’s of milliseconds, so this window cannot always be used for troubleshooting real time values such as motor current sampling, etc. Some other method such as streaming connection to the software may be required.

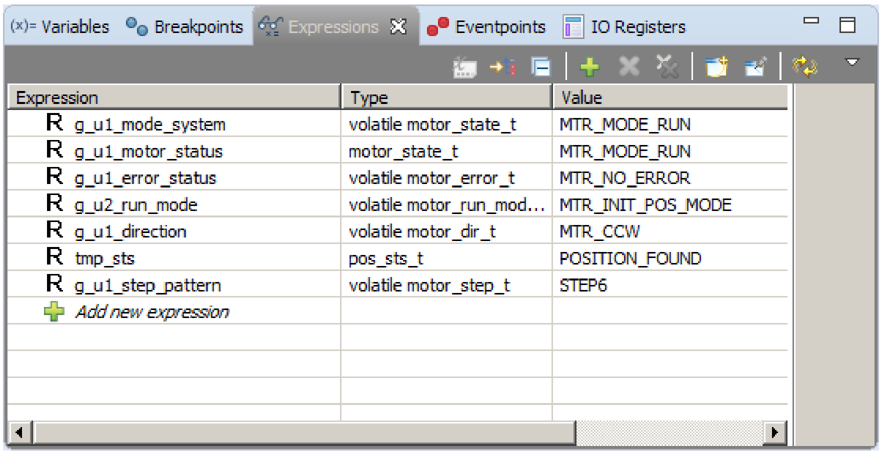

Figure 17: real time refresh Expressions

7.3.4 TODOs

Using e2Studio Task management feature, a number of comments have been marked with TODO text. These provide markers to places for "tuning and tweaking" to match a specific Motor/environment. Typically these will be numbers used to match a motor/systems performance, inertia, etc. for proper operation. TUNING MAY BE REQUIRED.

7.3.5 Doxygen

In the Doxy folder under the project, the CALLER/CALL graphs, definitions, variables, files structure, etc. can be browsed by clicking on the index.html file under the Project Explorer panel (or maybe opened outside in a conventional Web browser).

After changes are made to the software, you can update the Doxygen files by re-running the wizard and loading the Doxyfile in the Doxy directory.

8. References

1. RL78G1F Group User's Manual: Hardware (R01UH0516EJ0050)
2. R02AN0227EU0100 – Application Note: 6-Step Control of BLDC by Comparator Zero-cross detection.
3. R01AN2657EJ0100 - Application note: '120-degree conducting control of permanent magnetic synchronous motor: algorithm'
4. ROAR 2012 Rulesbook, http://www.roarracing.com/ (for sensor pin out on RC motor)
5. Applilet for RL78/G14 V1.01.01, used for sample TAU one-shot code generation
6. Trial series "T2001" 50W 60VA Low Voltage Inverter Unit User's Manual
7. http://www.electricaltechnology.org/2014/09/comparison-between-star-and-delta-connections.html

9. Appendix A

9.1 Current sampling Verification

Current sampling is critical to the operation of the Initial Position Algorithm. As we discussed earlier in the application note, the current sampling is a function of the ADC conversion rate and the time constant of the motor. We have looked at our ADC sampling time, so let’s look at how the motor time constant comes into play.

If you look at Figure 18, you can see that the motor current rises as a result of the active drive. The decay of current is also visible during the PWM off time. The current decay is a result of the R and L of the motor and the discharge paths through the inverter. In this case, the motor currents are re-circulating through two of the low-side MOSFETs, so the discharge path through the inverter should be relatively low in resistance (R of wire + R_{DSON} of MOSFET). You can see from the figure that it is alright to sample outside the active PWM, but if you delay too long it will affect the accuracy of your reading / analysis.

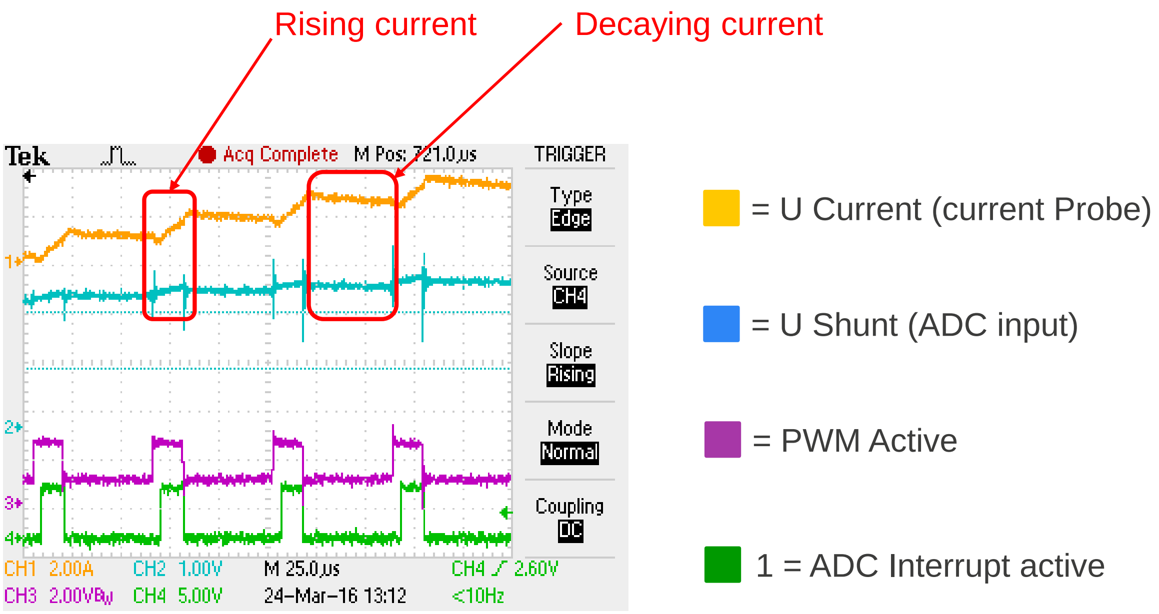

Figure 18: Current Sampling Detail

9.2 Y-Wound versus Delta-wound

Most of this application note is written in terms of a Y-Wound motor. The theory applies to Delta-wound motors as well since a mathematical relationship can be shown. The reader is left to make any mathematical calculations based on the Delta-to-Wye conversion.

https://en.wikipedia.org/wiki/Y-%CE%94_transform

9.3 Glossary

BEMF – Back Electromotive Force

BLDC Motor – brushless DC motor (typically requires electronic commutation)

DSO – Digital Signal Oscilloscope

MSO – Mixed Signal Oscilloscope

PM – Permanent Magnet

PWM - Pulse Width Modulation

TAU – Timer Array Unit

Website and Support

Renesas Electronics Website

http://www.renesas.com/

Inquiries

http://www.renesas.com/contact/

Revision History

		Description	
Rev.	Date	Page	Summary
1.0	06/24/2016		Initial Release

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 5.0