Constant air-flow control using iMOTION™ script language

iMOTION™ script language implementation

About this document

Scope and purpose

This document describes the application method of constant air flow control based on the algorithm implemented in iMOTION™ 2.0 script language. This is a functional extension of the existing “AN2020-20 Power calculation and constant-power control”. It is highly recommended for those who need the duct fan application by iMOTION™ system.

Intended audience

This document is intended for those who would like to use the script language option to implement the constant air flow control by PMSM.

Table of contents

About this document **1**
Table of contents **1**
1 Introduction **2**
2 Working characteristics of fan with ducts **3**
2.1 The characteristic curve of a fan 3
2.2 The characteristic curves of ducts 5
2.3 Combined operation of fan and ducts 6
3 Constant air flow control **9**
3.1 Introduction 9
3.2 Implementation 11
3.2.1 Establishing power versus speed curve 11
3.2.2 Variables for power control 15
3.2.3 Implementation 17
3.3 Test result 26
3.4 Limitation 29
4 References **30**
Revision history **31**

1 Introduction

The latest software release of iMOTION™ MCE includes the script language support, offering users the possibility to customize system level functionalities without affecting the motor and PFC control algorithm. The script language option enables additional control loops along with reading, writing and interfacing with the variables/parameters associated with the motor and PFC control.

In this document (AN2020-14), a constant air flow control by using script language code is introduced. This function is frequently used in centrifugal fan and pump motor control application, especially air conditioner, smoke exhaust ventilator and the duct fan application. It is also useful wherever controlling the cubic-volume air mass flow or the cubic-volume liquid mass flow are required.

A centrifugal fan application system often requires constant flow control when the resistance of air flow changes at given circumstances such as in a duct fan application. The resistance of air duct is often changed due to output fan diffuser angle adjustment, dust clogging, climate pressure and so on.

This document gives a method for providing constant air flow control for centrifugal fan application driven by a PMSM motor, which is based on the motor input constant power control. Previously a similar application note, “AN2020-20 Power calculation and constant-power control” was published, this document extends its control to more specific air flow volume control.

All the functions in this documents are implemented by script language code, and tested with the specific MADK reference design kit, EVAL-M1-CTF610N3 [1] and EVAL-M1-101T [2][3]. For more information about script language, please refer to the reference documents [4][5].

2 Working characteristics of fan with ducts

This chapter will introduce the operating characteristic of a centrifugal fan with duct and the factors influencing air flow so that a reader can understand the basic physics of the air flow.

2.1 The characteristic curve of a fan

The static pressure, dynamic pressure and total pressure of air or fluid in duct are introduced first. The static pressure in duct is the pressure of the air or fluid moving in the straight channel acting on the wall at a certain section is vertical, that is also to say as the air or fluid pressure to channel wall. The dynamic pressure in duct is the average pressure produced by the air or fluid flow velocity on the cross section, that can be also understood the motion energy of air or fluid. Figure 1 shows the static and dynamic pressure of inlet and outlet in centrifugal fan duct. As such, the static pressure is shown that the pressure to be measured at any point of vertical wall while the dynamic pressure which is the average air mass pressure in the duct and measured at any cross section of the duct. The P_{st1} and P_{dy1} are the static and dynamic pressure of inlet respectively; the P_{st2} and P_{dy2} are the static and dynamic pressure of outlet respectively. The total pressure in duct is the sum of static pressure and dynamic pressure, so the total pressure of inlet and outlet are P_{f1} and P_{f2}[9].

$$P_{f1} = P_{st1} + P_{dy1} \quad (1)$$

$$P_{f2} = P_{st2} + P_{dy2} \quad (2)$$

The total pressure of the centrifugal fan in Figure 1 is the energy increment of unit volume air or fluid passing through the fan, named as P_f, so P_f equals to the difference of outlet total pressure P_{f2} subtracting inlet total pressure P_{f1}[9].

$$P_f = P_{f2} - P_{f1} = P_{st2} + P_{dy2} - P_{st1} - P_{dy1} \quad (3)$$

The static pressure P_{st} of the centrifugal fan is the difference toltal pressure P_f of the centrifugal fan subtracting the dynamic pressure P_{dy2} of outlet. Note that it is not the static pressure P_{st2} of outlet[9].

$$P_{st} = P_f - P_{dy2} = P_{st2} - P_{st1} - P_{dy1} = P_{st2} - P_{f1} \quad (4)$$

So the static pressure P_{st} of the fan also can be viewed as the difference of outlet static pressure P_{st2} subtracting inlet total pressure P_{f1}.

The dynamic pressure P_{dy} of a centrifugal fan is the difference of the total pressure P_f of fan subtracting the static pressure P_{st} of fan[9].

$$P_{dy} = P_f - P_{st} = P_{dy2} \quad (5)$$

So the dynamic pressure P_{dy} of the fan is also the dynamic pressure P_{dy2} at the outlet.

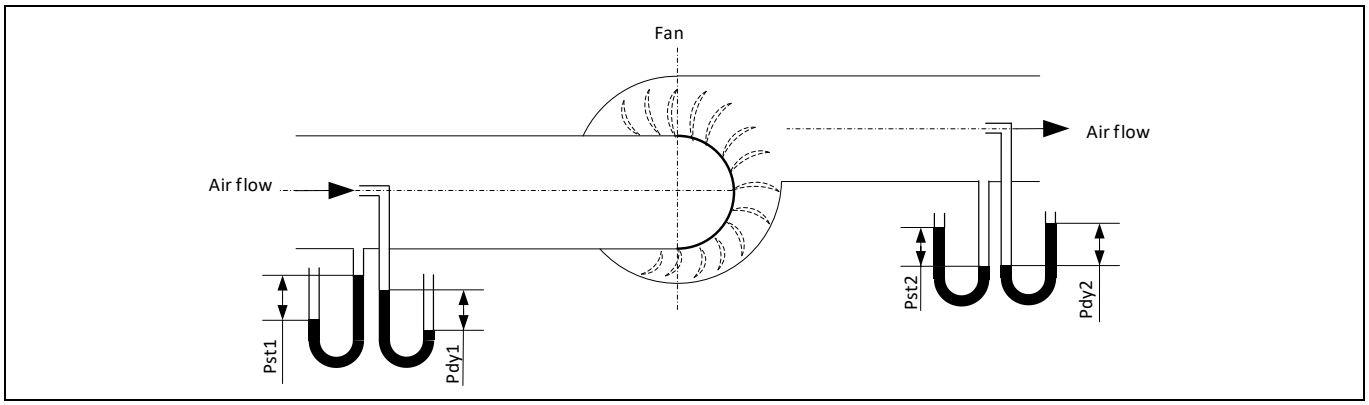

Figure 1 The static pressure and dynamic pressure of centrifugal fan duct

When dealing with a fan the key variables are pressure and air mass flow. Centrifugal fans are divided into forward type, backward type and radial type according to the outlet blade angle, but their real stable working P_f-Q (total pressure – flow) and P_{st}-Q (static pressure – flow) characteristic curves are described as Figure 2[9]. The total pressure and static pressure generated by the fan decreases as the air flow increase while the fan motor speed is kept constant. These curves will move up or down in parallel when the driving motor speed of the fan increases or decreases.

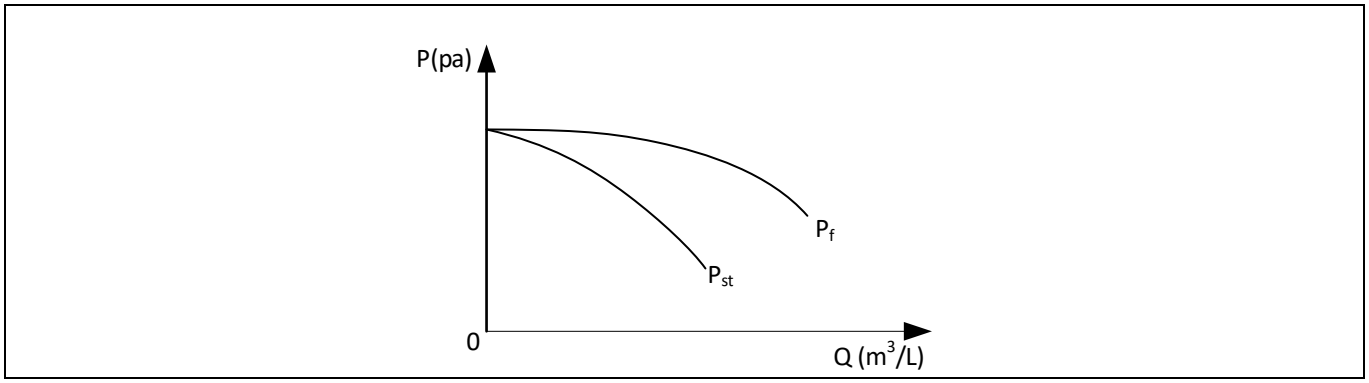

Figure 2 The P_f-Q and P_{st}-Q characteristic curves of centrifugal fan

The N-Q (power - flow) characteristic curve of these three types of centrifugal fans is described as Figure 3[9]. No matter forward, radial or backward centrifugal fan, their shaft power increases with the increase of air flow when all other operating conditions (duct resistance, climate pressure, temperature and so on) are constant. The output power N of fan is the product of total pressure P_f and flow Q.

$$N = P_f \times Q \ (kW) \qquad (6)$$

Because $P_f = P_{st} + P_{dy}$, so the power N also can be described as

$$N = P_{st} \times Q + P_{dy} \times Q(kW) = N_{st} + N_{dy} \qquad (7)$$

Where, N_{st}---- static pressure power; N_{dy}---- dynamic pressure power.

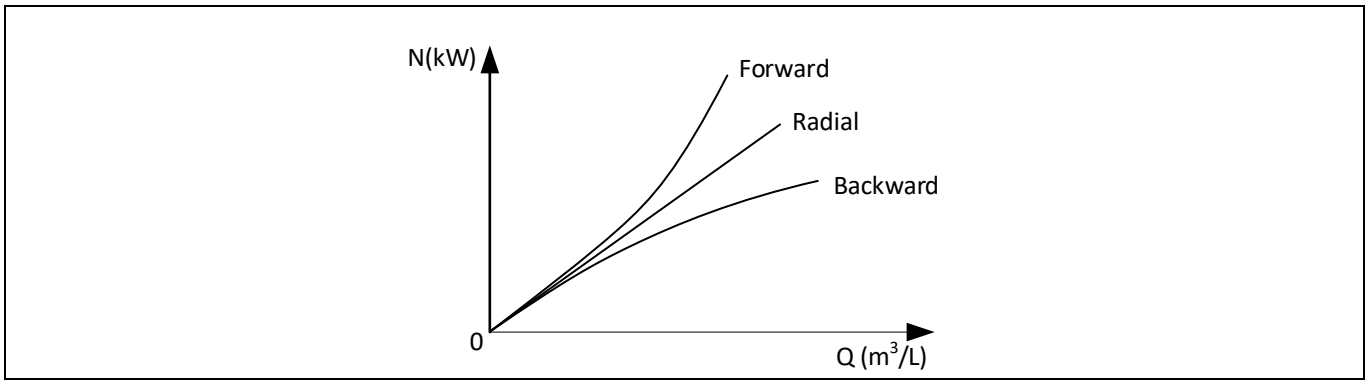

Figure 3 The N-Q characteristic curves of centrifugal

All the P_f-Q and N-Q curve are difficult to generate theoretically, so it is generally common that they are acquired by experiment.

2.2 The characteristic curves of ducts

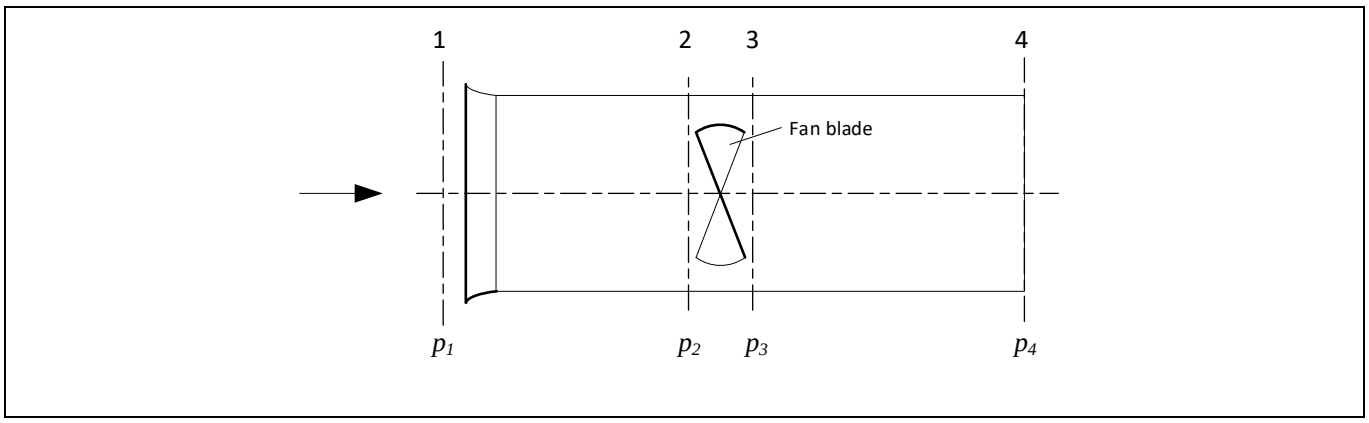

Figure 4 The ventilator ducts

Here the example of a duct air flow is examined as shown in Figure 4[10]. The entrance and exit of the duct are defined by the cross section p_1 and p_4. In this example the fan is located in-between "2" and "3" each cross-section points and the entrance and exit of fan are defined by the cross section p_2 and p_3.

1. The energy/pressure equation of cross section p_1 and p_2 is:

$$p_1 = p_2 + \frac{\rho}{2}v_2^2 + \Delta p_s \quad (8)$$

Where:
p_1 ----- Static pressure of cross section p_1 (N/m^2)
p_2 ----- Static pressure of cross section p_2 (N/m^2)
ρ ----- Fluid or air density in duct, assuming that the density doesn't change in all duct (kg/m^3)
v_2 ----- Average velocity of cross section p_2 (m/s)
$\frac{\rho}{2}v_2^2$ ----- Dynamic pressure of cross section p_2 (N/m^2)
Δp_s----- Pressure loss caused by entrance duct (N/m^2)

The pressure of cross section p_1 equals to p_a.which is the climate pressure outside the duct.

$$p_1 = p_a \quad (9)$$

2. The energy equation of cross section p_3 and p_4 is:

$$p_3 + \frac{\rho}{2}v_3^2 = p_4 + \frac{\rho}{2}v_4^2 + \Delta p_d \quad (10)$$

Where: Δp_d---- Pressure loss caused by duct exit (N/m^2)

The pressure of cross section p_4 is the climate pressure p_a.

$$p_4 = p_a \quad (11)$$

3. The fan generates the fluid or air total pressure difference between cross section p_2 and p_3. The total pressure P_f of the fan is the difference of entrance cross section p_2 total pressure P_{f2} and exit cross section p_3 total pressure P_{f3}.

$$P_f = P_{f3} - P_{f2} = (p_3 + \frac{\rho}{2}v_3^2) - (p_2 + \frac{\rho}{2}v_2^2) \quad (12)$$

$$P_f = \Delta p_s + \Delta p_d + \frac{\rho}{2}v_4^2 \quad (13)$$

Where P_f---- The total pressure generated by fan.

The above equation interprets that the total pressure generated by the fan is used to overcome the duct resistance and generate the speed of air or fluid.

Here the outlet dynamic pressure is $\frac{\rho}{2}v_4^2$, so the dynamic pressure generated by the fan is

$$P_{dy} = \frac{\rho}{2}v_4^2 \quad (14)$$

So static pressure generated by fan:

$$P_{st} = P_f - P_{dy} = \Delta p_s + \Delta p_d \quad (15)$$

The Δp_s and Δp_d are caused by the duct resistance and they are proportional to the square of flow Q, they can be described as:

$$\sum \Delta P_i = \Delta p_s + \Delta p_d = SQ^2 \quad (16)$$

Where $\sum \Delta P_i$ ---- The total resistance of the duct.
S ---- The coefficient of the fan's static pressure to square of duct' flow.

According to formula eq.(14), eq.(15) and eq.(16), the total pressure generated by fan rises and can be divided into two parts. One part is to overcome the duct resistance and that is said as static pressure P_{st}, the other part is the dynamic pressure P_{dy} which provides the dynamic power of the output air flow. So the static pressure P_{st} can be described as

$$P_{st} = \sum \Delta P_i = \Delta p_s + \Delta p_d = SQ^2 \quad (17)$$

We also can transform the eq. (13) to below equitation:

$$P_f = P_{st} + P_{dy} = \sum \Delta P_i + \frac{\rho}{2}v_4^2 = SQ^2 + \frac{\rho}{2}\left(\frac{Q}{F_d}\right)^2 = (S + \frac{\rho}{2F_d^2})Q^2 = FQ^2 \quad (18)$$

Where F_d ---- The cross coefficient of duct.
F ---- The coefficient of fan's total pressure to square of duct' flow.

Then the FQ^2 not only reflects the loss due to duct resistance, but it also includes the dynamic power.

According to eq. (17) and eq. (18) the characteristic curves of ducts total pressure of fan vs. duct's flow (P_f -Q) and static pressure of fan vs. duct's flow (P_{st} -Q) can be described as Figure 5.

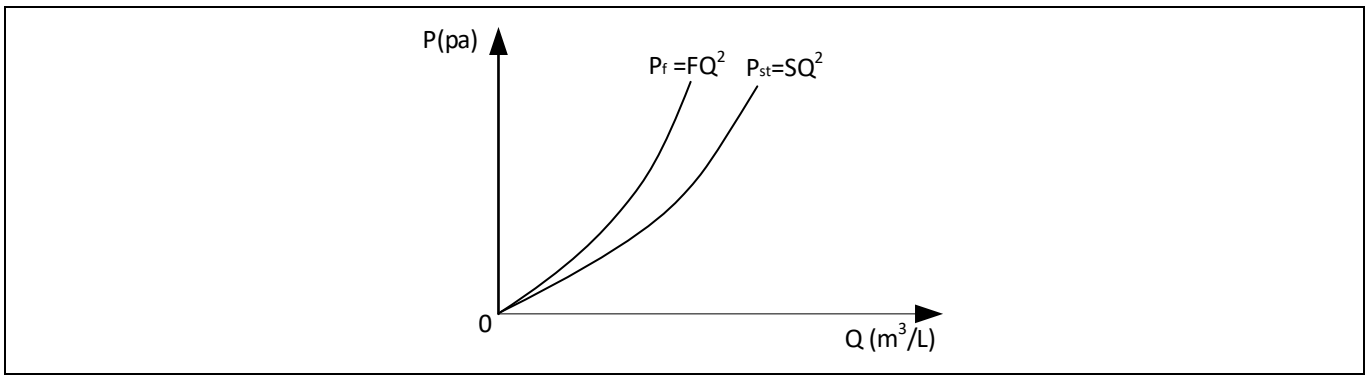

Figure 5 The characteristic curves of ducts

2.3 Combined operation of fan and ducts

Taking the forward centrifugal fan as an example, when drawing the P_f-Q (total pressure – flow) curve, the P_{st}-Q (static pressure – flow) curve of the fan (shown in Figure 2)and the characteristic curves of ducts total pressure

of fan vs. duct's flow (P_f -Q) and static pressure of fan vs. duct's flow (P_{st} -Q) (shown in Figure 5) into the same diagram, the stable working point A_f and A_{st} of fan with duct can be achieved.

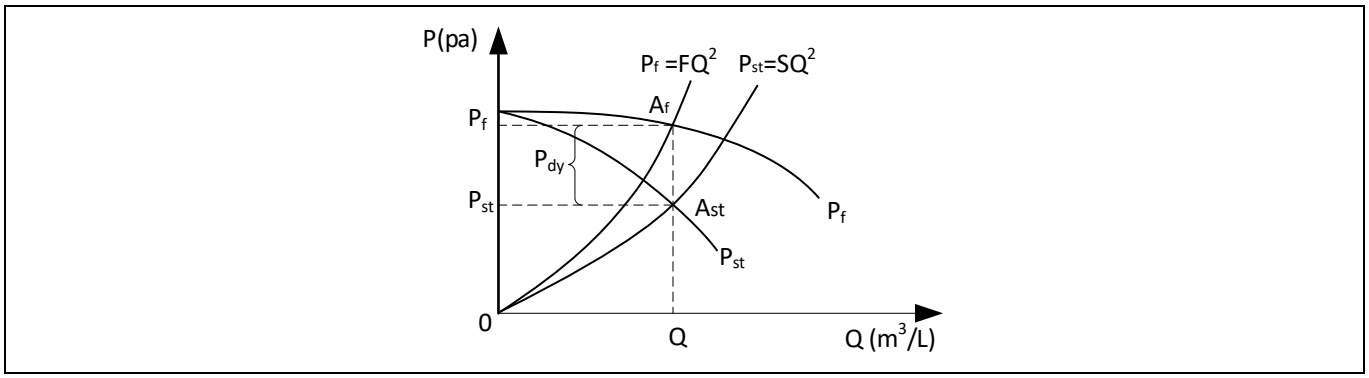

Figure 6 Stable working point of fan and ducts

As shown in Figure 6, when the driving motor speed of the fan is constant and ducts are fixed, then the air flow Q of the total system is given by the fan's characteristic and ducts' characteristic. A_f point represents the total pressure stable working point and A_{st} represents the static pressure stable working point. The difference between total pressure P_f and static pressure P_{st} is the dynamic pressure P_{dy}.

If the ducts are blocked or its resistance increases by other reasons and the driving motor speed dones't change, when the motor control mode is constant speed control, then the stable working point will change from A_f, A_{st} to B_f, B_{st} and the output air flow will change from Q_A to Q_B, as shown in Figure 7. The F_B is bigger than F_A due to the ducts' resistance increasing. The flow decreases from Q_A to Q_B , the total pressure increases from $P_{f\text{-}A}$ to P_{f_B}, and the static pressure of the fan increases from $P_{st\text{-}A}$ to $P_{st\text{-}B}$ for overcoming more ducts resistance. In inclusion, the total pressure and static pressure will increase and flow will decrease due to ducts resistance increase, but the ratio of flow decreasing is bigger than total pressure increasing, so the total power N of fan will decrease acorrding to $N = P_f \times Q$, then the driving motor output power will decrease.

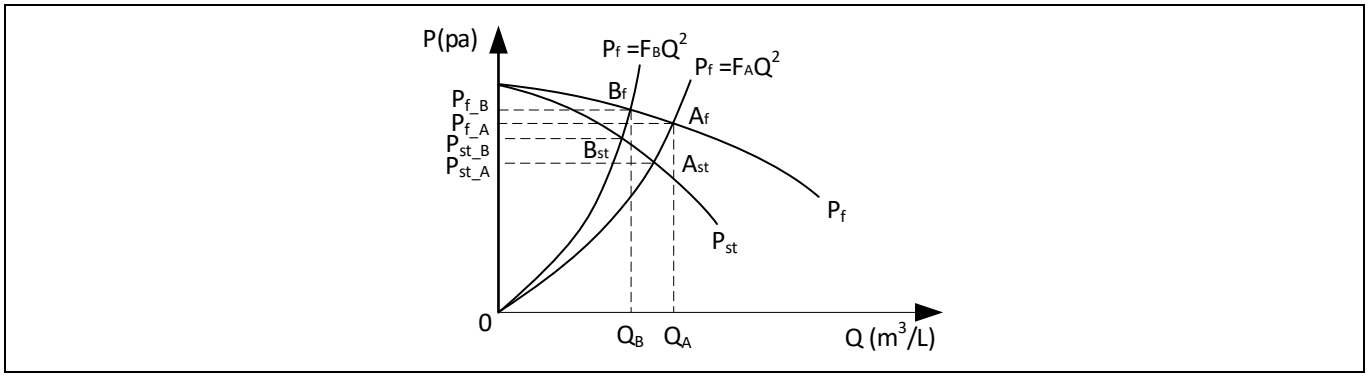

Figure 7 Working point changing by ducts' resistance increasing in motor constant speed control

If the driving motor uses constant power control, the motor will keep the output power constant. Provided that the motor is still constant speed control, the fan will work as shown in Figure 7, and the motor output power will decrease when the ducts' resistance increase. But when the motor control mode changed to constant power control mode, the motor will maintain its output power constant by increasing its speed, so the working point will change as Figure 8 shown. The Figure 8 only draws the total pressure vs. ducts' flow curve. The motor speed will increase from n_A to n_B for constant power control and the driver motor power will remain constant. But the output air flow Q_B is still smaller than previous flow Q_A, because although the motor output power is keept the same as before, the fan's static pressure increases for the ducts' resistance increased, so the static pressure power is more than the previous value in the same constant motor output power and the dynamic

pressure will be smaller than previous value. Then the output air flow still can't keep constant, but it is better than motor constant speed control mode.

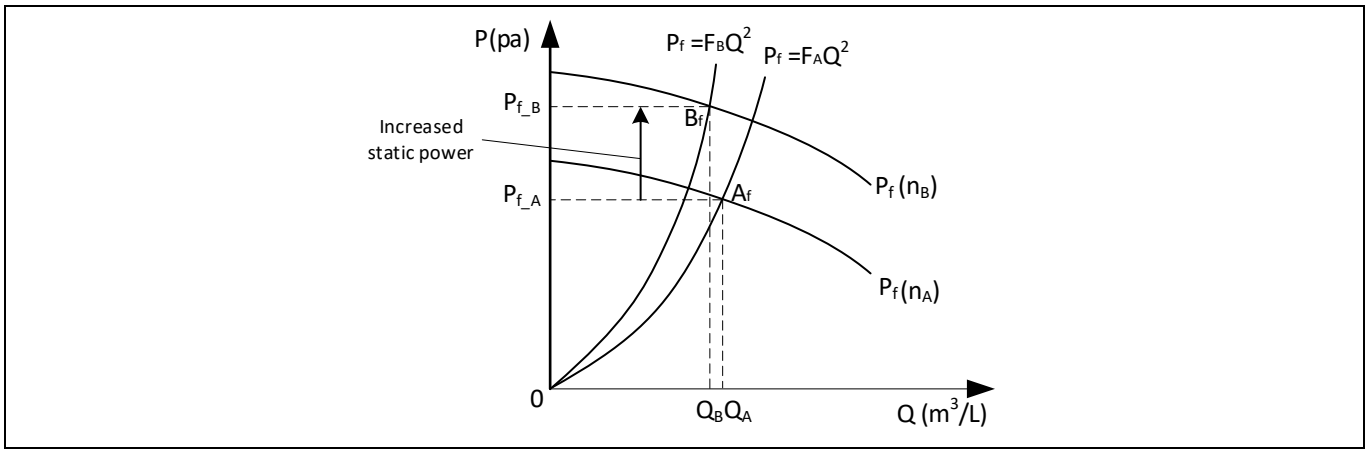

Figure 8 Working point changing by ducts' resistance increasing in motor constant power control

If a user wants to maintain the output air flow constant, the motor output power should overcome the added static pressure power and keep the dynamic pressure power, so the total power of driving motor output should increase.

This application will provide a type of constant air flow control method based on the motor power control. This method still uses power control method, but doesn't always keep the power constant, it will change the power target value according to the power-speed curve which maintains the output air flow constant.

3 Constant air flow control

3.1 Introduction

This chapter shows a method of constant air flow control which is implemented by the script language in the iMOTION™ IMC101T. This method is not limited to IMC101T, it is available to the whole iMOTION™ series products.

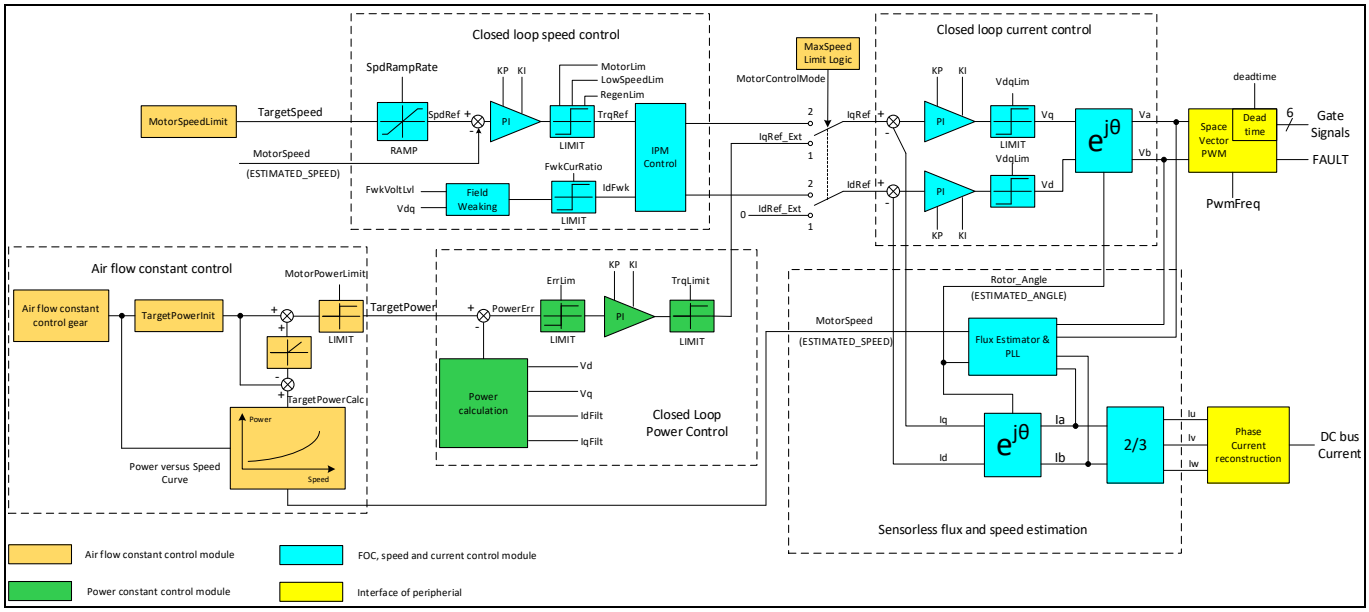

Figure 9 Constant air flow control block diagram

Figure 9 shows the block diagram of the constant power control. The yellow blocks are the interface of peripherals. The blue blocks are the basic algorithm existing iMOTION™ firmware which refer to iMOTION™ Motion Control Engine Software Reference Manual (2020) V1.3 [5]. The green blocks are the constant power control blocks which refer to Power_calculation_and_constant-power_control-ApplicationNotes[6]. The brown colored blocks are the constant air flow control which are the main items of this application. The MotorControlMode switches the outer loop control mode between the input of closed loop current control, the output of closed loop speed control and closed loop power control are for limiting for max motor speed with MotorspeedLimit. All the constant power and constant air flow control blocks are implemented by the script language.

The constant air flow control gear block is to select different TargetPowerInit values and provides the mapping of power versus speed table (curve). The power versus speed curve block is to calculate the corresponding power TargetPowerCalc satisfying constant air flow according to motor speed. This table or curve should be provided and acquired by test with the specific application duct and fan, and derived by constant power control prior to finalizing the control implementation. The final target power will feed the reference to the closed loop power control and is expressed in:

$$TargetPower = Limit_0^{MotorPowerLimit}\left(Limit_0^{+\propto}(TargetPowerCalc - TargetPowerInit) + TargetPowerInit \right) \quad (19)$$

Where: expression of $Limit_0^a x$ means limit the value of x from 0 to a.

Then the closed loop power control will implement power control according to the target power TargetPower. The difference between this method and constant power control is that the target power is a dynamic variable other than a static constant.

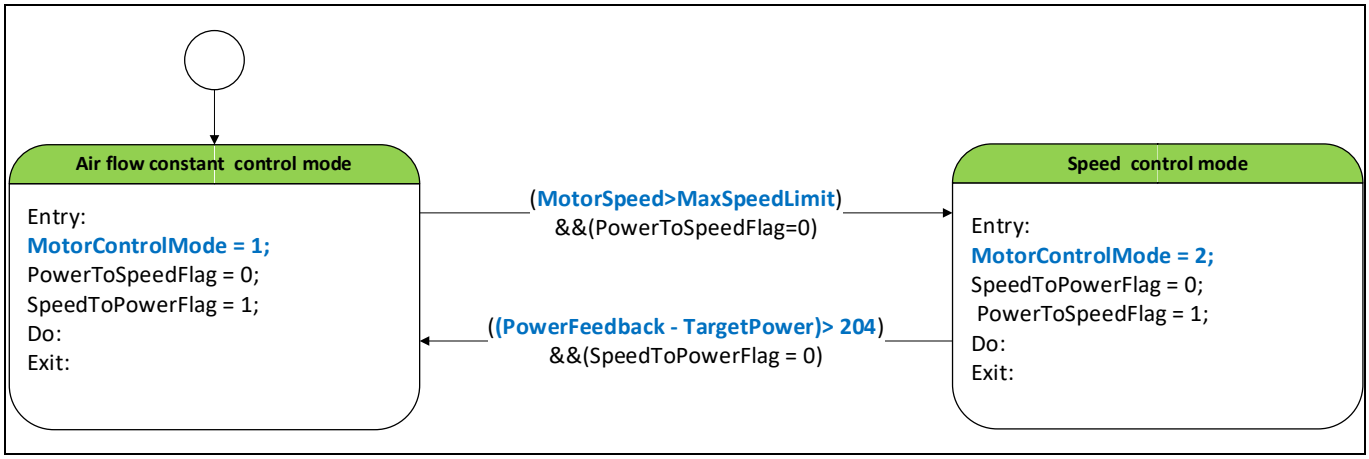

Figure 10 Max speed limiting logic

Figure 10 shows the maximum speed limiting logic which is for limiting the max speed of motor when power control is implemented. As is explained in chapter 2, the motor speed will increase when the ducts' resistance increases by constant power control, but the motor speed can't increase limitless, so the maximum speed must be limited.

In this method, the output of block MaxSpeed Limit Logic is MotorControlMode, as shown in Figure 9.

- MotorControlMode = 1: Constant air flow control mode
- MotorControlMode = 2: Speed control mode with Targetspeed equaling to max motor speed

The switching condition of constant air flow control mode and speed control mode is based on speed and power is shown in Figure 9.

- Control mode switches to speed control mode from constant air flow control mode when MotorSpeed is larger than MaxSpeedLimit.
- Control mode switches to constant air flow control mode from speed control mode when feedback power is larger 5%* rated power (4096 represent rated power) than target power.

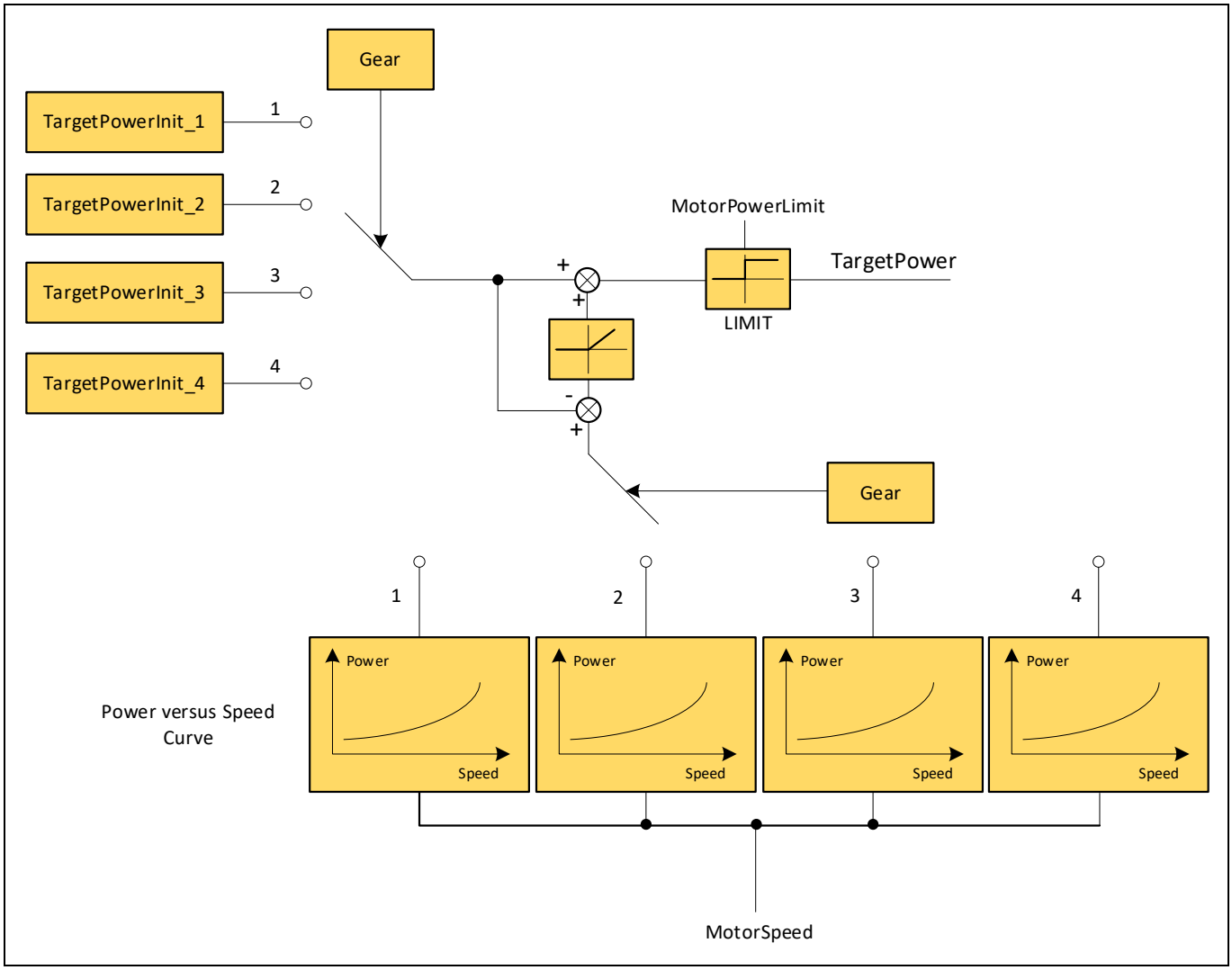

Figure 11 Multi-curve switching

Generally, there is not only one constant air flow gear in the application, there may be more constant air flow gears needed. As shown in Figure 11, that is the multi-curve selection logic. More power versus speed tables (curves) can be realized by tests at the beginning and its corresponding TagetPowerInit value.

3.2 Implementation

3.2.1 Establishing power versus speed curve

The power versus speed relationship needs to be established first before implementing constant air flow control. The test platform is shown in Figure 12. Here the air flow is measured with an air speed meter. The middle point air flow speed of the duct multiplied with the duct cross section area can represent the air flow Q passing the duct. If the measured air flow speed is constant, then the air flow Q can be regarded as constant.

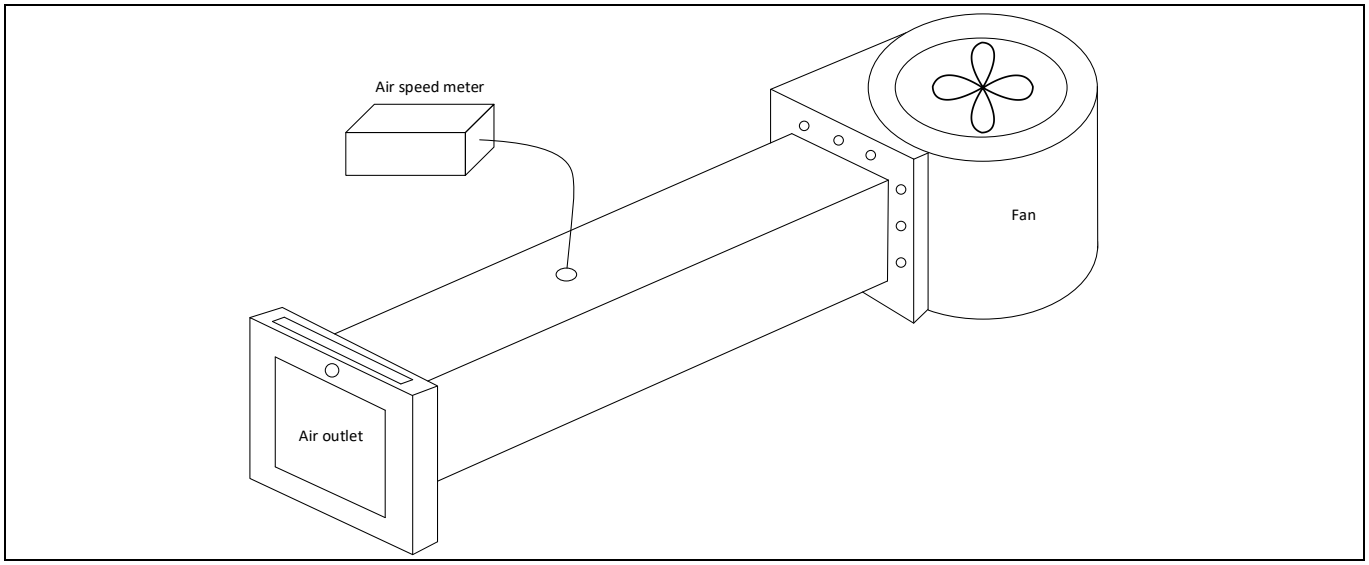

Figure 12 Constant air flow control test platform

The Table 1 is a test data example only with constant power control and multiple level of air blocking cases in the outlet with increments of 1/16 area of air outlet. If the motor input power is constant, the input power of the inverter changes little and the motor speed will increase, but the air flow speed decreases. Constant power control can't maintain the air flow speed constant, that is to say, it can't maintain the air flow Q constant. The Figure 13 shows the diagrams of test data with constant power control.

Table 1 Test data with constant power control

	Input power (W)	Target power (Scaled)	Motor speed (rpm)	Air flow speed (m/s)
No Blocked	16.3	400	448	6.6
1/16 Blocked	16.4	400	467	6.3
2/16 Blocked	16.45	400	476	6.1
3/16 Blocked	16.5	400	488	5.9
4/16 Blocked	16.5	400	501	5.7
5/16 Blocked	16.5	400	508	5.5
6/16 Blocked	16.55	400	517	5.25
7/16 Blocked	16.55	400	540	5.15
8/16 Blocked	16.6	400	580	4.6
9/16 Blocked	16.6	400	614	4.2
10/16 Blocked	16.6	400	635	3.6

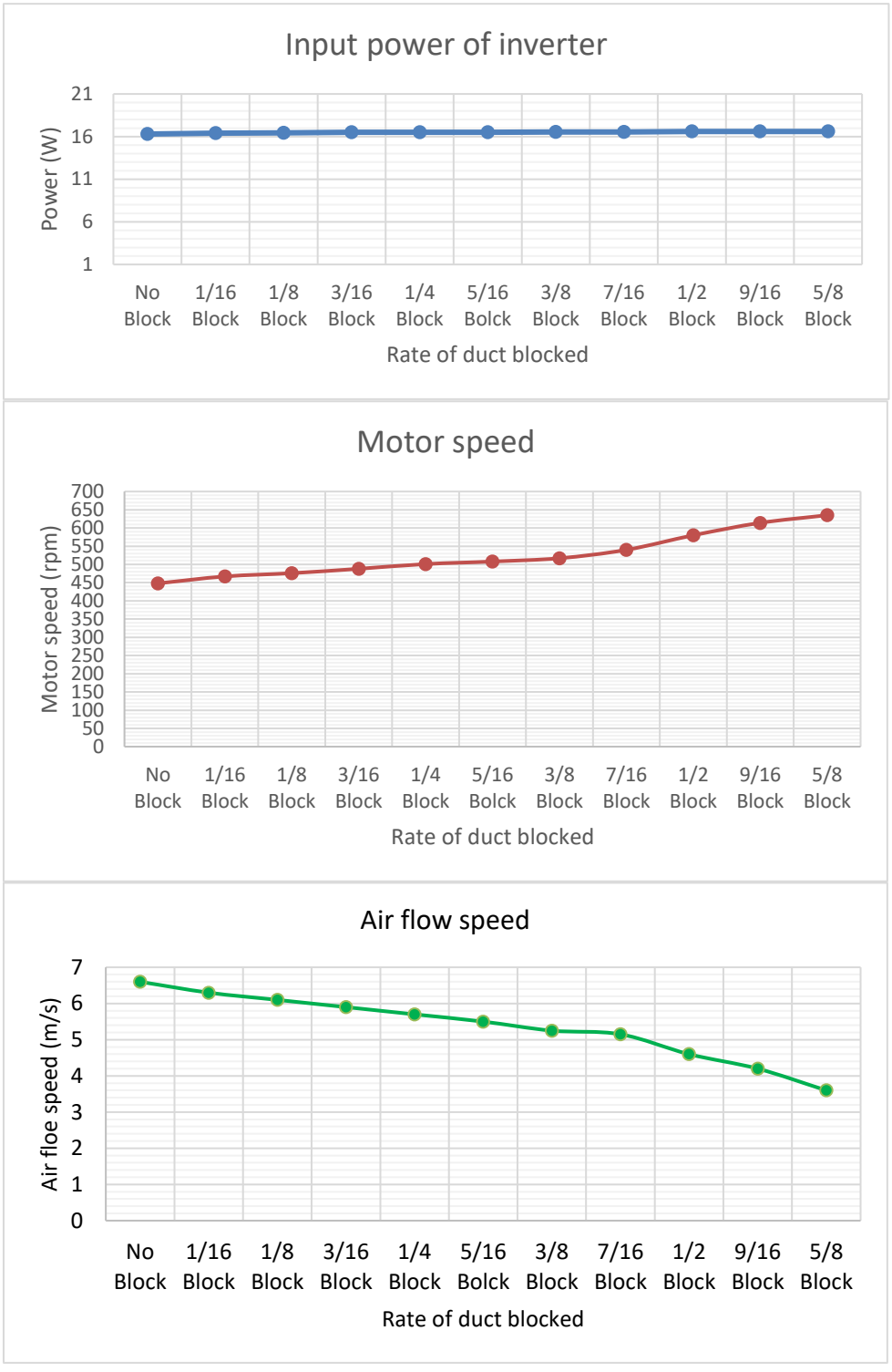

Figure 13 Diagram of test data with constant power control

If constant air flow should be obtained, then the motor power must be increased when the duct is blocked gradually. The Table 2 shows the test data with power control and air flow constant. In this control progress, in order to obtain constant air flow, we increase the target power of constant power control mode which has been implemented before when duct is blocked gradually, then we get the curve of target power versus motor

speed, all the target speed and motor speed values are given as real and scaled values. The curve is shown in Figure 14.

Note: *The scaled target power can be written in the MCEdesigner "Write Registers->Default->TargetPower". The scaled motor speed can be read in the MCEdesigner "Read Registers->Motor Speed Control->MotorSpeed". All the test process needs to be completed in the constant power control mode, please refer to application note Power_calculation_and_constant-power_control-ApplicationNotes[6].*

Table 2 Test data with power control and constant air flow

	Input Power (W)	Target power (Scaled)	Motor speed (rpm)	Motor speed (Scaled)	Air flow speed (m/s)
No Block	16.3	400	448	6672	6.6
1/16 Blocked	18.2	450	488	7268	6.6
2/16 Blocked	20.05	500	517	7700	6.6
3/16 Blocked	21.8	550	550	8192	6.6
4/16 Blocked	23.6	600	575	8564	6.6
5/16 Blocked	27.2	700	618	9204	6.6
6/16 Blocked	34	900	685	10202	6.6
7/16 Blocked	35.4	930	730	10873	6.6
8/16 Blocked	48.5	1300	877	13062	6.6
9/16 Blocked	55.5	1500	977	14551	6.6
10/16 Blocked	73	2000	1108	16503	6.6

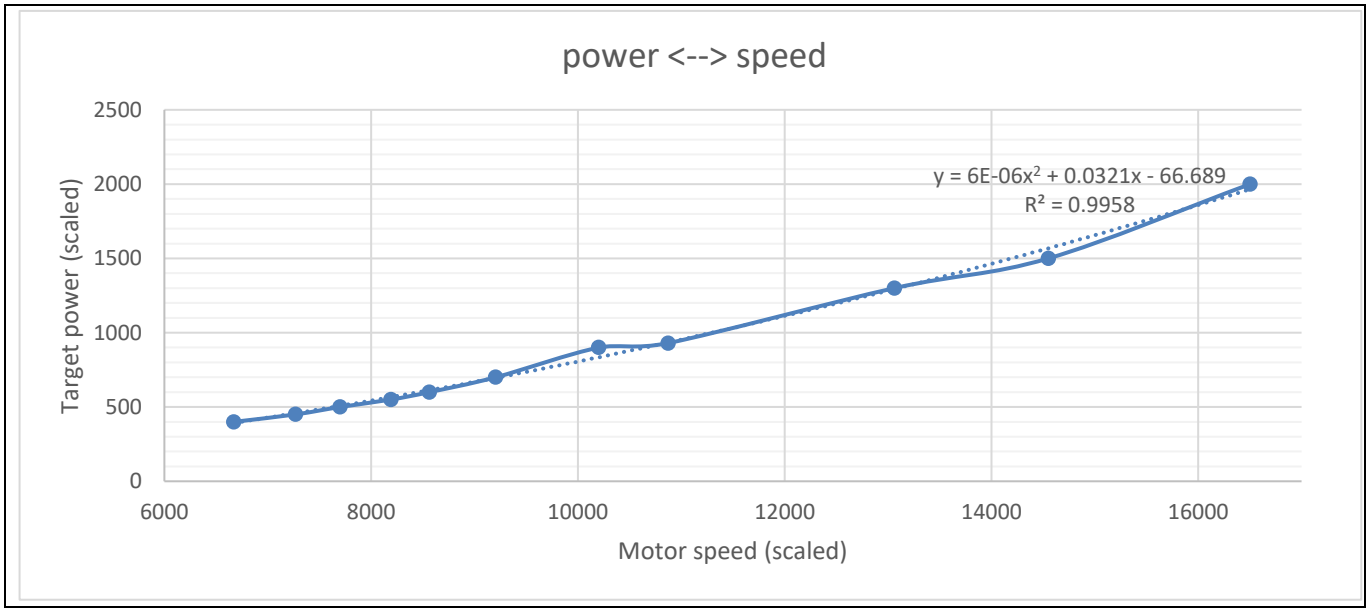

Figure 14 Power versus speed curve

The Figure 14 displays the power versus speed curve for constant air flow speed 6.6m/s, then a trendline is added to this curve and a polynomial is calculated. This polynomial is the final result needed. It is the key for achieving constant air flow control.

There are multiple ways to implement this power vs speed curve in script. One is to buid an equation matching this curve. Another way is to provide a look up table. These are common methods to implement the mapping of the desired curve. In this example, the equation-based implementation is used.

The polynomial has three coefficients, that are used to calculate the target power of inputting to motor according to the motor speed when duct is blocked as below formula in the script language code.

$$Target\ power = K1 \times Motor\ speed^2 + K2 \times Motor\ speed + K3 \quad (20)$$

In the Figure 14, $K1 = 6E-06$, $K2 = 0.0321$, $K3 = -66.689$.

For getting best air flow control accuracy, the three coefficients need to do appropriately rectification according to the smallest error calculated target power to the given target power. In this example, the final rectified coefficients are $K1 = 6E-06$, $K2 = 0.0321$, $K3 = -81$.

3.2.2 Variables for power control

This section introduces the parameters and variables in the demo code, explanation here can help you to read the demo code easier.

The demo code includes variables for power control and constant air flow control. The variables for constant air flow control will be introduced here, as for the variables for power control please refer to application note Power_calculation_and_constant-power_control-ApplicationNotes[6].

Note: If those parameters are defined as global variables in the script code, they can be written/read by MCEDesigner online when tuning the motor.

Global Variables definition:

TargetPowerInit

Scaling or notation	12 bits signed integer, 4095 = maximum motor power
Type	Variable, need to be tested before implementation
Description	This variable sets the initial input power of the motor, its value represents input power of motor when ducts output the set air flow Q without ducts blocked. This value should be tested with constant power control and without ducts blocked as described in section 3.2.1, and its value equals the TargetPower set for the set air flow Q in test. The maximum motor power is set by MaxPowerMulti100. TargetPowerInit is related to MaxPowerMulti100 setting. Take 100W motor for example, the register MaxPowerMulti100 need to be set to 100*100 = 10000, then 4095 represents 100W and 2048 represents 50W for TargetPowerInit.

Note: The Targetpower and MaxPowerMulti100 are the variables in the application note Power_calculation_and_constant-power_control-ApplicationNotes[6].

MotorPowerLimit

Scaling or notation	12 bits signed integer, 4095 = maximum motor power
Type	Variable, must be initialized in the script language code
Description	This register is to limit the maximum power of motor running. It must be initialized in the script language code, but can be changed in MCEdesigner.

MotorSpeedLimit

Scaling or notation	14 bits signed integer, 16383 = maximum motor speed
Type	Variable, must be initialized in the script language code
Description	This register is to limit the maximum speed of motor running. It must be initialized in the script language code, but can be changed in the MCEdesigner.

Gears

Scaling or notation	16 bits signed integer. Gears for air flow Q by user setting, such as 1, 2, 3…
Type	Variable
Description	This variable for user to change different power versus speed curve, then change the constant air flow Q gears. This value can be changed by communication of GPIO.

Local variabls in task0

MotorControlMode

Scaling or notation	16 bits signed integer.
Type	Variable, must be initialized in the script language code.
Description	1- Power control mode (default value). 2- Speed control mode. This value to switch power control mode and speed control mode automatically. When motor speed is over maximum speed then control mode change to speed control mode from power control mode. when feedback power is 5%* maximum motor power larger than target power.

PowerToSpeedFlag

Scaling or notation	16 bits signed integer.
Type	Variable, must be initialized in the script language code.
Description	0- Power control mode (default value) 1- Speed control mode. This variable is auxiliary for power control mode and speed control mode automatically switch. Don’t need to change in running process.

SpeedToPowerFlag

Scaling or notation	16 bits signed integer.
Type	Variable, must be initialized in the script language code.
Description	0- Speed control mode (default value) 1- Power control mode. This variable is auxiliary for power control mode and speed control mode automatically switch. Don’t need to change in running process.

Local variabls in task1

K11

Scaling or notation	32 bits signed integer, Q24.
Type	Variable, must be initialized in the script language code.
Description	This is the first coefficient of power versus speed curve polynomial

K21

Scaling or notation	32 bits signed integer, Q12.
Type	Variable, must be initialized in the script language code.
Description	This is the second coefficient of power versus speed curve polynomial

K31

Scaling or notation	32 bits signed integer, Q0.
Type	Variable, must be initialized in the script language code.
Description	This is the third coefficient of power versus speed curve polynomial

If the user needs to add more power versus speed curves, then more variables could be added. Take the second curve for example, the variables K12, K22, K32 could be added.

3.2.3 Implementation

The script language program consists of the following part [5]:

- Set Commands: Define script user version and script task execution period.
- Functions: script code should be written inside four predefined function, which are Script_Task0_init (), Script_Task0 (), Script_Task1_init () and Script_Task1 ().
- Variables and parameters.
- Statement and expressions: Each individual statement must end with a semicolon.
- Comments: Starts with a slash asterisk /* and ends with an asterisk slash */ for multiple line comments, starts with double slash // for single line comments.

The script engine supports 2 independent tasks, namely, Task 0 and Task 1, running concurrently. The user script program runs repeatedly on a configurable interval within Task 0 or Task 1 loop. The shortest possible execution period is 1 ms for Task 0, and 10 ms for Task 1. The execution period for each task can be configured to the multiples of 1 ms for Task 0 or 10 ms for Task 1 in the script language code. Task 0 has higher priority than Task 1. Task execution period can be configured using “SCRIPT_TASK0_EXECUTION_PERIOD” and “SCRIPT_TASK1_EXECUTION_PERIOD” in the set commands section. Other parameters corresponding the task execution period need to configure are “SCRIPT_TASK0_EXECUTION_STEP” and “SCRIPT_TASK1_EXECUTION_STEP”, they represent the number of lines to be execute every task’s shortest possible execution period which are 1ms and 10ms separately.

The power control function in this demo code is running in the Task 0 since the minimum execution period of Task 1 is 10 ms, and this long period will introduce too much delay which will cause bad dynamic response and even unstability. The power versus speed curve calculation is running in the Task 1. Take this demo code for example, “SCRIPT_TASK0_EXECUTION_PERIOD” sets 2 and “SCRIPT_TASK0_EXECUTION_STEP” sets 50, which means that the execution period for this task is 2 ms and 50 lines of instructions are executed every 1 ms. “SCRIPT_TASK1_EXECUTION_PERIOD” sets 1 and “SCRIPT_TASK0_EXECUTION_STEP” sets 30, which means that the execution period for this task is 10 ms and 30 lines of instructions are executed every 10 ms.

There are some factors which need to be considered in order to determine those settings.

1. Function minimum loop time

Loop time need to make sure the function run normally. Take LPF design for example, according to Nyquist theorem, the sampling frequency needs to be at least higher than twice of the frequency of interest to realize

effective attenuation, so choosing the sampling period 1 ms for the LPF would be effective for the frequency ranging up to 500 Hz, and if sampling period is 2 ms, the frequency will down to 250 Hz.

In this demo, function works normally when the “SCRIPT_TASK0_EXECUTION_PERIOD” is set to 1~5, but the less period is setting, the faster the dynamic response is. So 2 is used in the demo since 1 will consume more CPU resource.

2. Numbers of instructions

The following Code Listing 1 shows a portion of the compiled script object file (.ldf) where it shows at line 008 that the number of instructions for Task 0 is 48. So the “SCRIPT_TASK0_EXECUTION_STEP” should be set greater than 48 to ensure the entire loop of Task 0 is completed during each minimum execution period 1 ms, since the “SCRIPT_TASK0_EXECUTION_PERIOD” in this demo is 2, “SCRIPT_TASK0_EXECUTION_STEP” set greater than half of the 48 is feasible so that entire loop of Task 0 is completed during execution period 2 ms, first 1 ms script engine executes the specific lines of the script code and the rest of code is done in the next 1 ms. In the demo code, “SCRIPT_TASK0_EXECUTION_STEP” sets 50.

In the same way, for task 1, “SCRIPT_TASK0_EXECUTION_STEP” is set to 30.

Code Listing 1 Portion of compiled script object file for constant air flow control

```
001     %-------------------------------------------------------------
002     % Script Object File
003     %-------------------------------------------------------------
004     % SCRIPT_USER_VERSION              : 001.000
005     % DATE & TIME                      : 01.02.2021  18:33:06
006     % SIZE                             : 1167 Bytes
007     % Total Number of Lines            : 294
008     % Task0 - Number of Instructions   : 48
009     % Task1 - Number of Instructions   : 18
```

3. CPU Load

The CPU resource is prioritized for the implementation of the motor and PFC control algorithm. The script engine is designed to take advantage of the spare CPU resource for the execution of the script program. The priority of the execution of the script program is lower than that of the motor and PFC control algorithm, so that it won’t affect the performance of the control algorithm. However, CPU usage needs to be carefully evaluated before enabling the script function. “SCRIPT_TASK0_EXECUTION_PERIOD” should meet application/function time requirement, and it must be ensured that the CPU load is not exceeding 95%, otherwise the script function will become abnormal.

There are 2 methods to evaluate the CPU load, detail of which can be found in the section 3 of the reference[4],

The script engine supports 2 kinds of 32-bits unsigned integer variables which are global variables and local variables. The maximum number of global variables supported by the script engine is 30, and the maximum number of local variables for each task is 24. The communication between Task 0 and Task 1 can be realized by using global variables. Only global variables are accessible from MCEDesigner or user UART interface. It is recommended to define a variable as the global type if users intend to read its value during run time using MCEDesigner [8].

The variables defined in the lines 009-022 of the Code Listing 2 are global variables, below are the instructions on how to import them to the MCEDesigner if you need to write/read them by it.

a) After the MCEWizard compiled the script code, a map file will be generated in the folder with the suffix “.map”.

b) Open the MCEDesigner file whose suffix is ".irc", import the map file by File→Import map file as Figure 15 shown.

c) Then the defined global variables could be found in both Read/Write registers in the motor1 window and as source of the trace setup in the system window. The detail is shown in the Figure 16.

Note: The variables in the MCEDesigner will divide into low 16 bits with suffix "_L" and high 16 bits with suffix "_H" since the MCEDesigner only support 16 bits integer.

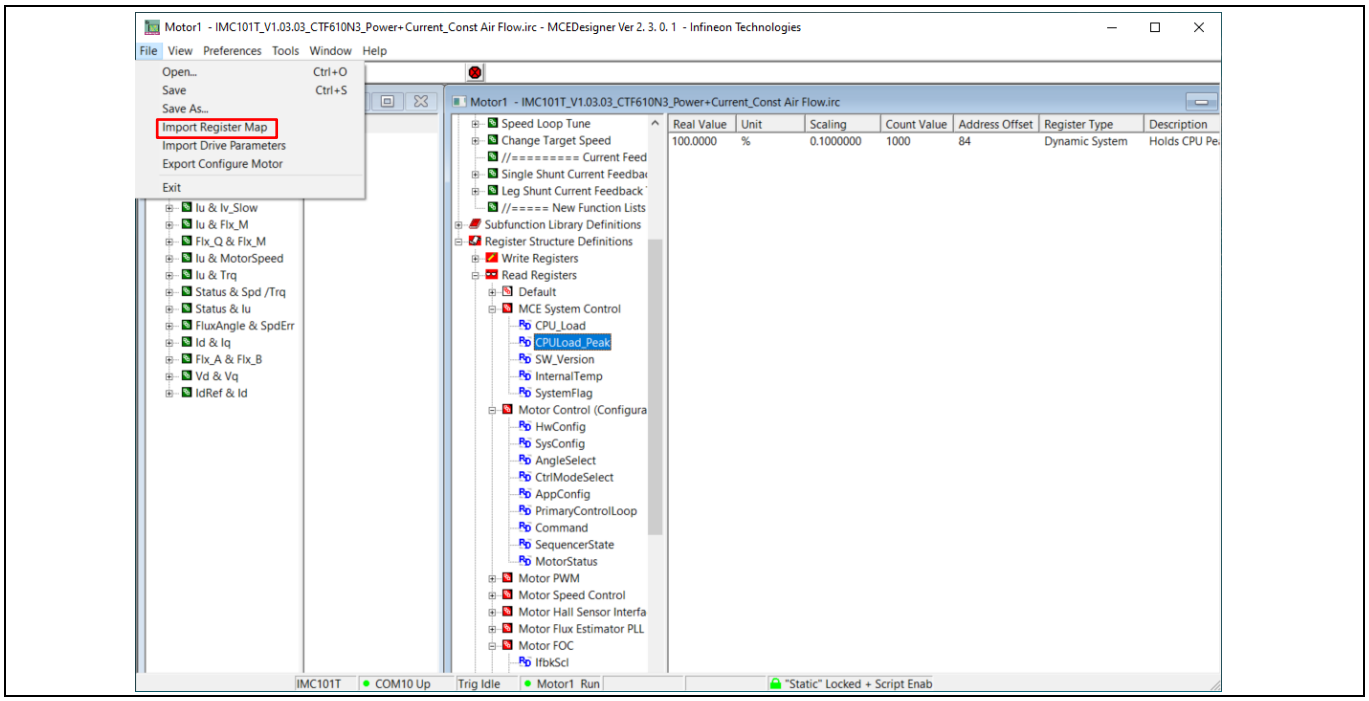

Figure 15 Import the register map by MCEDesigner

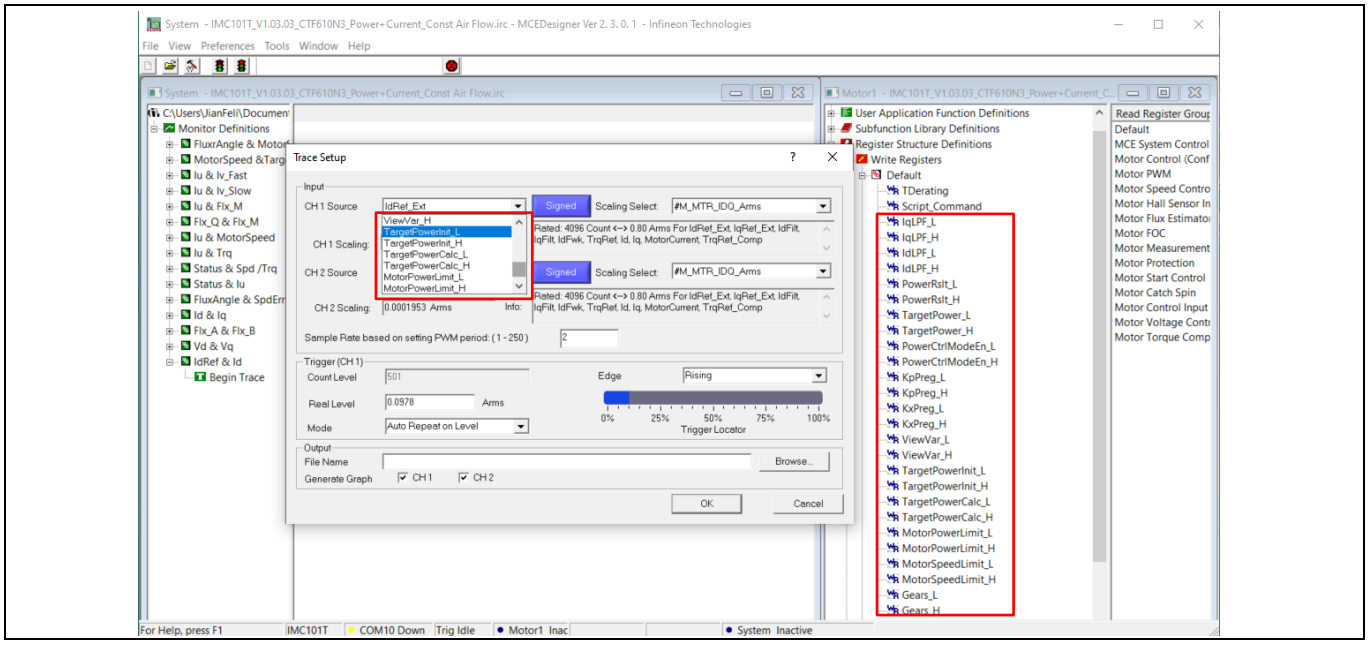

Figure 16 Global variables in the MCEDesigner

4. Demo code

Code Listing 2 lists the demo code with script for constant air flow control.

The variables defined in the initializing function are local variables, which can only be read/written by defined Task itself. Line 029–049 in the demo code are local variables definition, and Line 051–084 are variables initialization for task 0. Line 225 – 235 in the demo code are local variables definition, and Line 237 – 256 are variables initialization for task 1. Both global and local variables can be initialized here, and both Script_Task0/1_init () functions are only called once after startup.

The functions Script_Task0/1() are the script functions which are executed periodically by the set period.

In the function Script_Task0(), line 094 – 101 are the power calculation with d/q axis voltage and current. LPF in the lines 099 - 100 are commented out since the phase lag of LPF will affect the stability of the power control, if you only use the power calculation without power control, you can add the LPF here to make the power result more stable. Lines 106 – 124 are the conditions to judge if to start or stop the motor, DC bus voltage is chosen to set the flag. Users can set the condition based on their own requirements. Lines 126 - 218 are constant air flow control code. Line 129 is to calculate the target power dynamically. Lines 132- 135 are the maximum power limitation. Lines 140 – 158 are the switch of speed limitation mode with power control mode. Lines 160 - 210 are the code of the power control section. There are 3 sub-sections, first is the error calculation and limitation, using the variable ErrLim to limit the power error so that the power could ramp up with the specific slope. Next is the PI control section of power controller, the Integral part is limited by variable IntegralLim. The last section limits the control output and set the reference torque of the current controller. Lines 212 – 217 are the speed limitation code.

In the Script_Task1_init (), the power-speed curves of constant air flow running are defined. More different curves can be defined and two of the same curves are defined in the demo code.

In the function Script_Task1(), the target power TargetPowerCalc for power control is calculated by the power – speed curve dynamically here. Different curves can be selected by different Gears value.

Code Listing 2 Demo of the constant air flow control

```
001       #SET SCRIPT_USER_VERSION (1.00)  /*Script version value should
   be 255.255*/
002       #SET SCRIPT_TASK0_EXECUTION_PERIOD    (2)  /*Script execution
   time for Task0 in mS, maximum value 65535*/
003       #SET SCRIPT_TASK1_EXECUTION_PERIOD    (1)  /*Script execution
   time for Task1 in 10mS, maximum value 65535*/
004       #SET SCRIPT_START_COMMAND (0x3)  /* Start command, Task0 :
   Bit0, Task1 : Bit1; if bit is set, script executes after init */
005       #SET SCRIPT_TASK0_EXECUTION_STEP (50)  /* Script Task0  step,
   This defines number of lines to be executed every 1mS*/
006       #SET SCRIPT_TASK1_EXECUTION_STEP (30) /* Script Task1  step,
   This defines number of lines to be executed every 10mS*/
007       */
008       /*************************************************************/
009       /*Global Variables Definition*/
010       int IqLPF,IdLPF;
011       int PowerRslt;//1 = 0.01W
012       int TargetPower; //Q12,4096 means max power
013       int PowerCtrlModeEn;
014       int KpPreg,KxPreg;   //Q12
015       int ViewVar;
016
```

Code Listing 2 Demo of the constant air flow control

```
017        /*Air flow constant control variables*/
018        int TargetPowerInit; //Q12,4096 means max power
019        int TargetPowerCalc; //Q12,4096 means max power
020        int MotorPowerLimit; //Q12,Limit the max power of motor; 4096
   means rated max power.
021        int MotorSpeedLimit; //Q14,limit the max speed of motor; 16383
   means rated max speed.
022        int Gears;           //Airflow gear for different airflow
   constant control.
023
024
025        /*************************************************************
   ************************************************/
026        /*Task0 init function*/
027        Script_Task0_init()
028        {
029         /*Local Variable definition*/
030         /*Power Calculation variable*/
031          int IqMultiDEN,PwrMultiDEN;
032          int TempVar;
033          int DPwr,QPwr;
034          int PowerScl;
035
036         /*Constant power control variables*/
037          int TargetTrqTemp,TrqLimit;
038          int Power_Inetgral;
039          int MaxPowerMulti100;
040          int PowerQ12;
041          int PowerErr;
042          int InetgralLim;
043          int ErrLim;
044          int TimeCnt;
045
046          /*Air flow constant control variables*/
047          int MotorControlMode;
048          int PowerToSpeedFlag;
049          int SpeedToPowerFlag;
050
051          /*Power Calculation variable initialize*/
052          //3/2*Irated*sqrt(2)*2048/4307*4096/sqrt(3)/DC feedback
   scale*100
053               PowerScl =23272;
054          // take GA for example, PowerScl =
   INT(1.5*0.8*1.414*2048/4307*4096/sqrt(3)/8.2*100)=23272
055          // 8.2 means DC bus feedback scale is 8.2 counts/V
056          // *100 so the Power result resolution is 0.01W
057          // you can adjust this parameter to get the correct power
   result
058
059          /*Constant power Control variables intializing*/
060          MaxPowerMulti100 = 15000; //150W
061          ErrLim = 200;
062          TargetPower = 0;
063          KpPreg = 3200;                 //3200
```

Code Listing 2 Demo of the constant air flow control

```
064          KxPreg = 800;                    //800
065          PowerCtrlModeEn = 1;
066          //CtrlModeSelect = 1;   //1- current mode 2- speed mode
067          TrqLimit = 4096;         //4096 means 100% Torque
068
069          /*Air flow constant control variables intializing*/
070          TargetPowerInit = 400;
071          MotorPowerLimit = 3277;  //Q12, Limit the max power of
   motor; 4096 means rated max power=150W, limit to 3277--120W
072          MotorSpeedLimit = 14000; //Q14, limit the max speed of
   motor; 16383 means rated max speed.
073          MotorControlMode = 1;    //1--Power control mode  2--Speed
   control mode.
074
075
076          /*Constant power Control variables intializing to zero*/
077          IdRef_Ext = 0;
078          TimeCnt = 0;
079          Power_Inetgral = 0;
080          PwrMultiDEN = 0;
081
082          /*Air flow constant control variables intializing to zero*/
083          PowerToSpeedFlag = 0;
084          SpeedToPowerFlag = 0;
085
086        }
087        /*************************************************************
   ************************************************/
088        /*Task0 script function*/
089        Script_Task0()
090        {
091          //=======================================
092          // P = 3/2*(VdId+VqIq)
093          //=======================================
094          DPwr = (IdFilt*Vd)>>12; //Q12
095          QPwr = (IqFilt*Vq)>>12; //Q12
096
097          TempVar = (PowerScl * (QPwr+DPwr))>>12;
098        //LPF Ts 1ms (2.5Hz -3db)
099        //  PwrMultiDEN= PwrMultiDEN + (TempVar - PowerRslt);
100        //  PowerRslt = PwrMultiDEN >> 6;
101         PowerRslt = TempVar;
102
103         /*Constant Power Control*/
104         //monitor the DC bus voltage, if it is higher than the level,
   motor will start.
105
106          if (VdcFilt > 1500)
107          {
108                  TimeCnt = TimeCnt + 1;
109                  if(TimeCnt > 500)
110                  {
111                        Command = 1 ;  //start the motor
112                        PowerCtrlModeEn = 1; //
```

Code Listing 2 Demo of the constant air flow control

```
                        TimeCnt = 501;

                }

            }
            else
            {
                    Command = 0;
                    TimeCnt = 0;
                    PowerCtrlModeEn=0;
                    Power_Inetgral=0;
            }

            if(PowerCtrlModeEn == 1)
            {
                  //calculate the target power
                  TargetPower = TargetPowerInit + TargetPowerCalc;//Q12

                  //limit TargetPower to maxPower permitted.
                  if(TargetPower>MotorPowerLimit)
                  {
                      TargetPower = MotorPowerLimit;
                  }

                  //calculate the power feedback
                  PowerQ12 = (PowerRslt<<12)/MaxPowerMulti100; //to Q12

                  //Maxspeed limitation and restoring
                  if((MotorSpeed > (MotorSpeedLimit))&&(PowerToSpeedFlag
   == 0))
                  {
                       //limit motor speed to MotorSpeedLimit
                       MotorControlMode = 2;
                       SpeedToPowerFlag = 0;
                       PowerToSpeedFlag = 1;
                  }

                  if(MotorControlMode == 2)
                  {
                      if(((PowerQ12-TargetPower)>
   204)&&(SpeedToPowerFlag == 0)) //if PowerQ12 is larger than
   TargetPower 5% rated power, then switch to MotorControlMode 1
                      {
                            //restore to power control mode
                            MotorControlMode = 1;
                            PowerToSpeedFlag = 0;
                            SpeedToPowerFlag = 1;
                      }
                  }

                  //Power control
                  if(MotorControlMode == 1)
                  {
                     CtrlModeSelect = 1; //1- current mode 2- speed mode
```

Code Listing 2 Demo of the constant air flow control

```
164
165                     //calculate the power error
166                     PowerErr = TargetPower-PowerQ12; //Q12
167
168                             //limit error input
169                             if(PowerErr>ErrLim)
170                             {
171                                     PowerErr = ErrLim;
172                             }
173                             else
174                             {
175                                     if(PowerErr < (0-ErrLim))
176                                     {
177                                             PowerErr = 0-ErrLim;
178                                     }
179                             }
180
181                             //PI
182                             InetgralLim = 1<<24;//limit 2^24
183                             Power_Inetgral =
   Power_Inetgral+PowerErr*KxPreg;
184
185                             if(Power_Inetgral>InetgralLim)
186                             {
187                                     Power_Inetgral=InetgralLim;
188                             }
189                             else
190                             {
191                                     if (Power_Inetgral<0 )
192                                     {
193                                             Power_Inetgral = 0;
194                                     }
195                             }
196                             TargetTrqTemp =
   (PowerErr*KpPreg+Power_Inetgral)>>12;//convert to Q12
197                             //Limit trqref,you could also limit it by
   motorlim
198                             if (TargetTrqTemp > TrqLimit)
199                             {
200                                     TargetTrqTemp = TrqLimit;
201                             }
202                             else
203                             {
204                                     if(TargetTrqTemp<0)
205                                     {
206                                             TargetTrqTemp = 0;
207                                     }
208                             }
209                             IqRef_Ext = TargetTrqTemp;
210                 }
211
212                 //Speed limitaition control
213                 if(MotorControlMode == 2)
214                 {
```

Code Listing 2 Demo of the constant air flow control

```
                    CtrlModeSelect = 2;//1- current mode 2- speed mode
                    TargetSpeed = MotorSpeedLimit;
                }
          }
        }

        /*****************************************************************************************************/
        /*Task1 init function*/
        Script_Task1_init()
        {
          /*Air flow constant control variables*/
          int K11;
          int K21;
          int K31;

          int K12;
          int K22;
          int K32;

          int temp1, temp2, temp3;
          int TimeCnt2;

          /*Air flow constant control variables intializing*/
          Gears = 1;

          //Curve1
          K11 = 100;  /* Q24, 0.000006*2^24=100*/
          K21 = 131;  /* Q12, 0.0321*2^12=131*/
          K31 = 81;

          //Curve2
          K12 = 100;  /* Q24, 0.000006*2^24=100*/
          K22 = 131;  /* Q12, 0.0321*2^12=131*/
          K32 = 81;


          /*Air flow constant control variables intializing to zero */
          TargetPowerCalc = 0;
          TimeCnt2 = 0;
          temp1 = 0;
          temp2 = 0;
          temp3 = 0;
        }
        /*****************************************************************/
        /*Task1 script function*/
        Script_Task1()
        {
              //power-speed curve 1
              if(Gears == 1)
              {
                    TargetPowerInit =400;

                    temp1 = (K11 * MotorSpeed)>>12;
```

Code Listing 2 Demo of the constant air flow control

```
268                     temp2 = (temp1 * MotorSpeed)>>12;
269                     temp1 = (K21 * MotorSpeed)>>12;
270                     temp3 = temp2 + temp1 - K31;
271                     TargetPowerCalc = temp3 - TargetPowerInit;
272                     if(TargetPowerCalc < 0)
273                     {
274                           TargetPowerCalc = 0;
275                     }
276             }
277
278             //power-speed curve 2
279             if(Gears == 2)
280             {
281                     TargetPowerInit =(400 * 2867)>>12;
282
283                     temp1 = (K12 * MotorSpeed)>>12;
284                     temp2 = (temp1 * MotorSpeed)>>12;
285                     temp1 = (K22 * MotorSpeed)>>12;
286                     temp3 = temp2 + temp1 - K32;
287                     TargetPowerCalc = ((temp3 * 2867)>>12) -
   TargetPowerInit;
288                     if(TargetPowerCalc < 0)
289                     {
290                           TargetPowerCalc = 0;
291                     }
292             }
293
294       }
```

3.3 Test result

The Figure 17 shows the dynamic process of power when the duct's outlet in the Figure 12 is blocked. It can be seen that the speed and power of motor increase when the outlet is blocked and decrease when the outlet is unblocked agian.

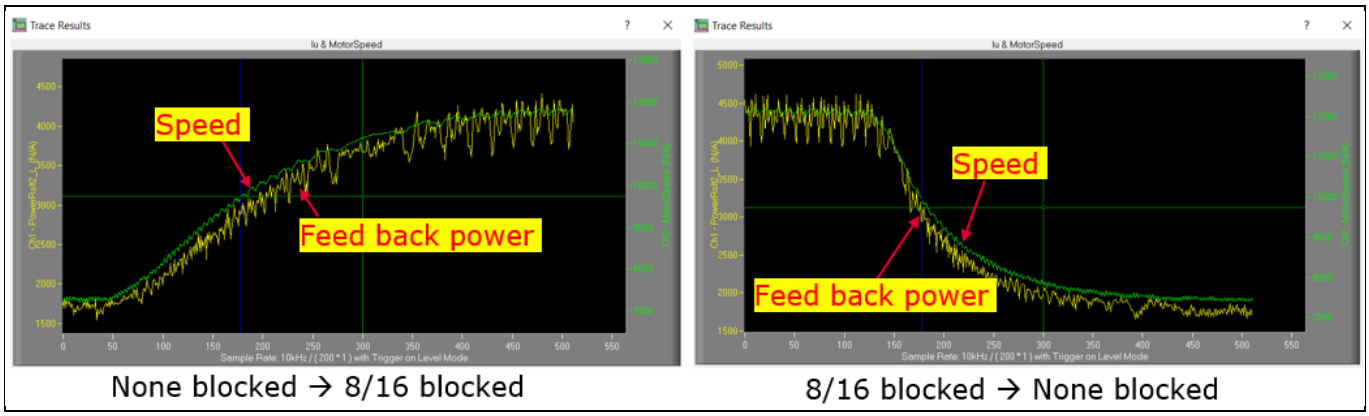

Figure 17 The dynamic process of power when duct's outlet is blocked

The Figure 18 shows the target power and feedback power in the dynamic process of operation in Figure 17. The target power is calculated by the motor speed dynamically in the none blocked to blocked or blocked to none blocked process. It can be seen that the feedback power can follow the target power well.

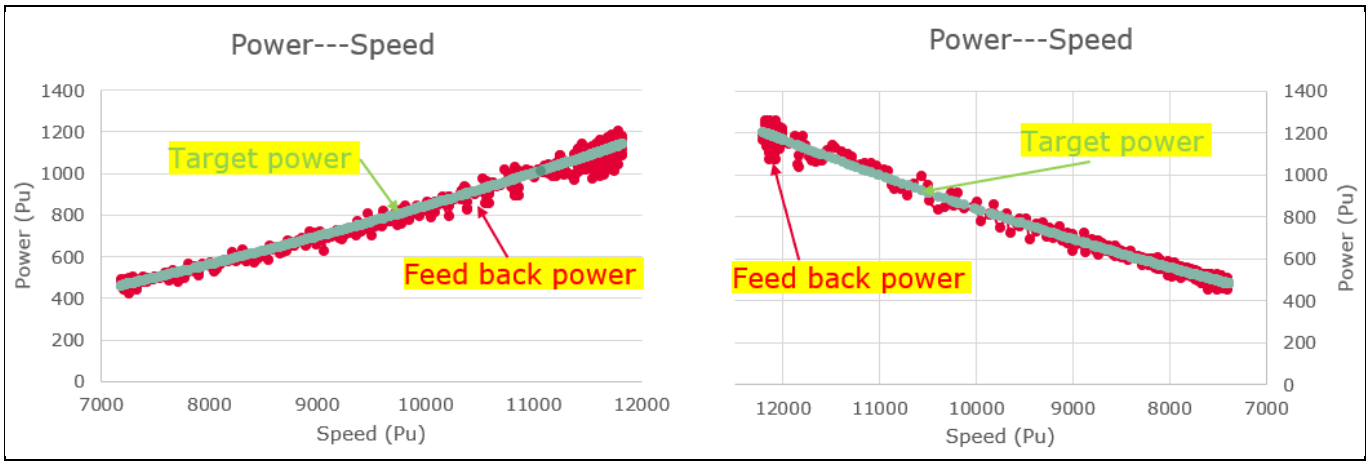

Figure 18 The target power and feedback power in dynamic process

The Figure 19 shows the air flow speed measured in the duct in the Figure 12 when the outlet of duct is blocked many times and the degree of blocking each time is different. It can be seen that the air speed can return to the constant value 7.2 m/s whenever the disturbance appears, that means the air flow Q is constant.

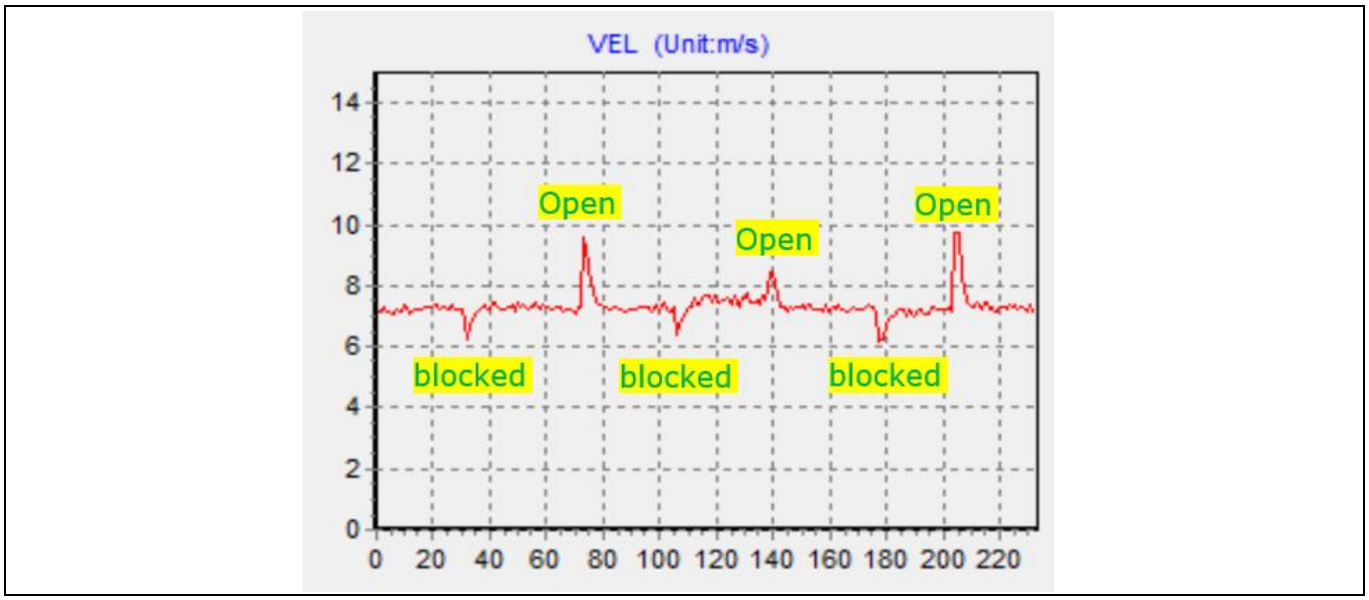

Figure 19 The air flow speed when duct's outlet is blocked many times

Figure 20 shows the air flow speed measured in the duct in the Figure 12 when the outlet of duct is blocked one time and the reaction of the motor output current. It can be seen that the motor output current changes smoothly and the air speed can keep 7.0 m/s. It is slightly different from the Figure 19 test result, because the ambient temperature here is lower than Figure 19's ambient temperature.

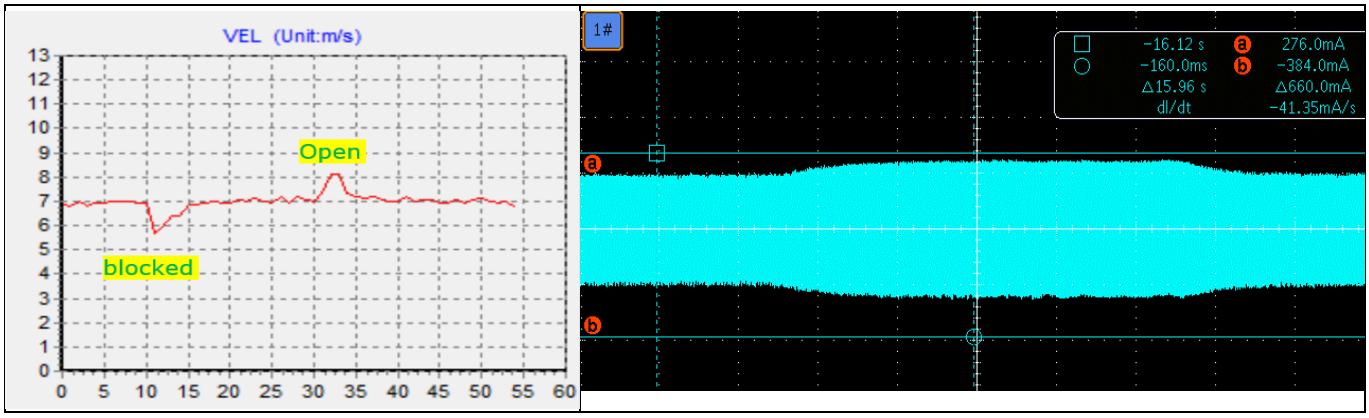

Figure 20 The air flow speed when duct's outlet is blocked one time and the motor output current

Figure 20Figure 21 shows the air flow speed measured in the duct in the Figure 12 when the outlet outside air pressure changes randomly and the reactions of the motor output current. It can be seen that air speed can be kept constant and the motor output current changes smoothly.

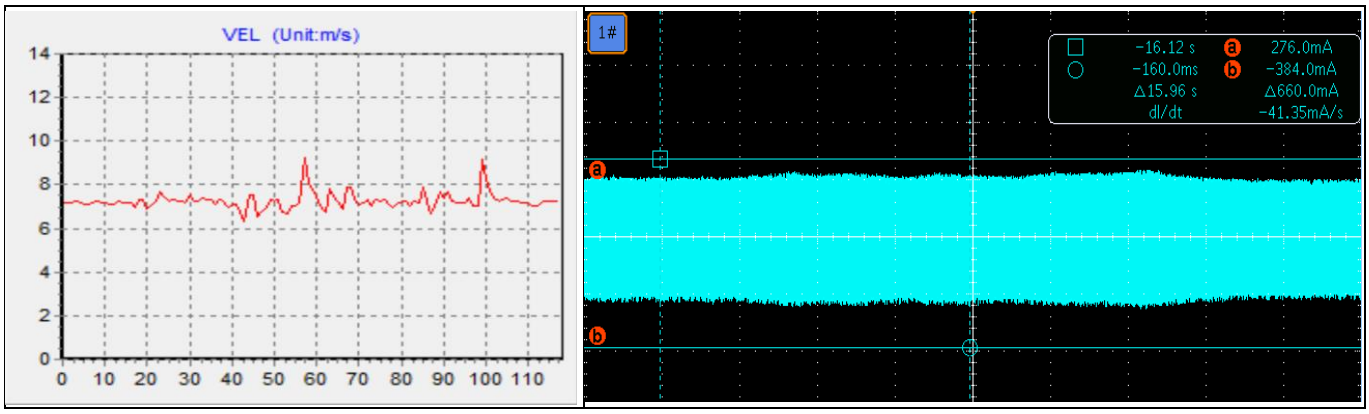

Figure 21 The air flow speed when duct's outlet is blocked one time and the motor output current

Figure 22 shows the air flow speed measured in the duct in the Figure 12 when the outlet is blocked and the degree of blocking is too much resulting in motor speed exceeding the maximum speed allowed. Then the constant air flow control algorithm can limit the maximum speed. In the speed limitation process the air flow can't be kept constant, but the air flow will return the set value when the degree of blocking decreases.

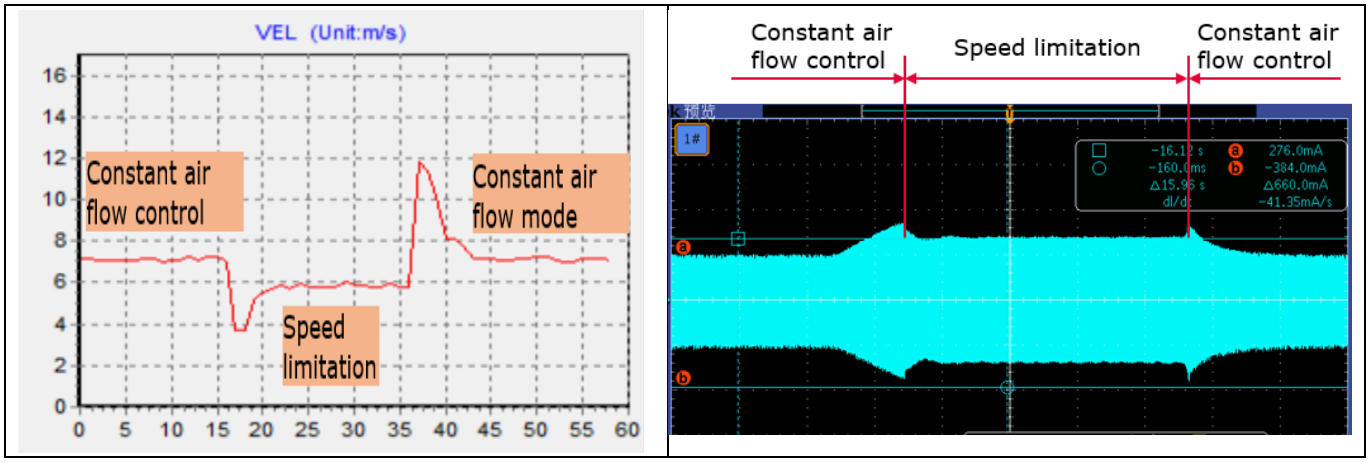

Figure 22 Maximum speed limitation

3.4 Limitation

The following items need to be considered when the constant air flow control is applied.

1. This method is only available to centrifugal fan.
2. This method is not available to SVPWM over modulation condition but available to flux weakening. In the SVPWM over modulation state, it becomes inaccurate for measuring feedback motor power.
3. The tested air flow constant value Q will be influenced by air temperature, humidity and atmospheric pressure, but the humidity influence is only small, so it can be ignored. All the influence factors will change the density of the air, then influence the constant air flow value. If the air density becomes smaller than the power-speed curve tested conditions, then the actual constant air flow value will be bigger than the tested value. If the air density becomes bigger than the power-speed curve tested conditions, then the actual constant air flow value will be smaller than the tested value.

The relationship between air density, temperature, altitude is as below:

$$\rho' = \rho_0 \frac{273}{273 + t} \times \frac{p}{0.1013}$$

ρ'---- The air density at centigrade temperature t ℃ and absolute atmospheric pressure p, (kg/m^3).

ρ_0---- The air density at centigrade temperature 0 ℃ and standard atmospheric pressure $0.1013MPa$, $(\rho_0 = 1.293kg/m^3)$.

p---- The absolute atmospheric pressure altitude, (MPa).

t---- The centigrade temperature, (℃).

Then user can utilize the below formula to correct air flow:

$$Q_a = \sqrt{\frac{\rho_a}{\rho_T}} Q_T$$

Q_T---- The air flow tested (m^3/s).

ρ_T---- The air density of Q_T tested condition (kg/m^3).

Q_a---- The actual air flow (m^3/s).

ρ_a---- The actual air density (kg/m^3).

4 References

[1] Infineon Technologies AG.EVAL-M1-CTE610N3 User Manual V1.0

[2] Infineon Technologies AG.EVAL-M1-101T User Manual V1.6

[3] Infineon Technologies AG. Datasheet of Infineon IMC101T-T038 (2019). V1.4

[4] Infineon Technologies AG. How to use iMOTIOM™ Script language.(2018) V1.0

[5] Infineon Technologies AG. iMOTION™ Motion Control Engine Software Reference Manual (2020) V1.3

[6] Infineon-AN2020-20_Power_calculation_and_constant-power_control-ApplicationNotes-v01_00

[7] Infineon Technologies AG. MCEWizard_V2.3.0.0 User Guide (2019)

[8] Infineon Technologies AG. MCEDesigner_V2.3.0.0 Application Guide (2019)

[9] 龚光彩, 流体输配管网[M], 3 版, 北京: 机械工业出版社, 2004. 11-12, 30-32, 303-308.

[10] 周明, 风机的恒风量控制系统[D], 杭州: 浙江大学, 2007.

Revision history

Document version	Date of release	Description of changes
V 1.0	2021-02-03	Initial release.
V 2.0	2022-01-28	1. Modify the grammar errors. 2. Modify the title of this article.
V 2.1	2022-04-13	Add two references.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2022-4-13

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2022 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
AN-2021-14

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

Please note that this product is not qualified according to the AEC Q100 or AEC Q101 documents of the Automotive Electronics Council.

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.