# Flying Saw – JM-2xx



Jetter AG
Gräterstrasse 2
D-71642 Ludwigsburg
Germany

| | |
|---|---|
| Phone - Switchboard: | +49 (0)7141/2550-0 |
| Phone - Sales: | +49 (0)7141/2550-530 |
| Phone - Technical Hotline: | +49 (0)7141/2550-444 |
| Fax: | +49 (0)7141/2550-425 |
| E-mail - Sales: | sales@jetter.de |
| E-mail - Technical Hotline: | hotline@jetter.de |
| Internet Address: | http://www.jetter.de |

# Table of Contents

# 1 Introduction

This application note describes programming a JetMove 2xx for the "Flying Saw" application. In this case, the JetMove 2xx is driven as two single axes and not in connection with the Motion Control (MC). For better understanding of this application note, the Application Note 37 "Flying Shear – General Information" should be read beforehand.

In this Application Note, definite steps for implementing various applications of the Flying Saw, as have been mentioned in Application Note 37, are described. Instructions on implementation can be found in the following chapters:

- Classical application of the Flying Saw, as of chapter 4 "Classical Flying Saw“
- Flying Saw with hardware signal as of chapter 5 "Flying Saw with Hardware Signal“

Concerning additional demands made on the range of Flying Saw functions, please refer to chapter 6 "Further Required Features“.

The programming of individual variants is illustrated by respective programming examples. These programming examples have been collected in the JetSym-ST project "JM2xx_FS_ST_Example". The project can be downloaded from the download area of the Jetter AG homepage.

For programming, the JetSym-ST function library for the Flying Saw function is needed. It consists of the ST program files LibJM2xxFS_h.stp and LibJM2xxFS.stp. These files contain structure definitions and functions needed for implementing the applications. The files can also be downloaded from the download area of the Jetter AG homepage.

The function library has been described in detail as of chapter 9 "Function Library "Flying Saw“.

For all example program parts which have been listed in this documentation, the variable declaration given in chapter 9.7 "Standard Variable Declaration for All Sample Programs“ on page 37 is binding.

## 1.1 Prerequisites

The Flying Saw function is implemented by means of the JM-2xx table function. The functions of the library take over the handling of the table function, which means that the user has only got to know the table function roughly. The only thing the user has to know is how to set up a table in JM-2xx. This information can be taken from the document "jm2xx_at_jetcontrol_bi_xxxx_user_information.pdf".

In order to use the table function, an axis group including motion master and motion slave must also be configured. Information on configuring a axis group can also be taken from the document "jm2xx_at_jetcontrol_bi_xxxx_user_information.pdf", chapter "Master-Slave Mode". For configuring an axis group, there are motion instructions provided by JetSym.

### 1.1.1 System requirements

The following software versions are needed as a minimum:

| Software Versions | |
|---|---|
| **Motion Control / Modules / Development Environment** | **Software Version** |
| JetSym | V. 3.0 |
| JetMove 2xx | V. 2.07 |
| JX2-CNT1 | V. 3.0 |

## 1.2 Restrictions

The functions of the library have been designed for the following situations:

- Both motion master and sawing axis are travelling in positive direction
- The saw axis starts the sawing cycle out of standstill
- The sawing axis carries out an optimum synchronising phase; this means that – while synchronising - the sawing axis does not make up for distance, in order to reach the cutting position; it rather starts off early. Regarding the time-optimised synchronising phase, please turn to Application Note 37.

# 2 Program Initialisation

## 2.1 Steps and Sequence of the Initialisation

For using the function library, the following functions must be processed within the initialisation range of the control program in the set order:

- Carry out the MotionFS_Ini function for initialising the Flying Saw structure for all sawing axes available
- Carry out the functions MotionFS_LoadTab_Linear, respectively MotionFS_LoadTab_Sin2 for calculating and loading the required tables
- Initialising the axis groups in Master-Slave Mode

See example project in the task t_Init

## 2.2 Setting the Master Positioning Range in the Slave

When initialising an axis group, the master positioning range of the sawing axis must also be set. The master positioning range is set in the sawing axis via axis register R158 "Max. Master Position" and R159 "Min. Master Position". Setting the master positioning range of the sawing axis correctly is essential for correct behavior of the application. The following must be given heed to:

The master positioning range should never become greater than 250,000 mm. For rotatory axes, 360° is always set.

We recommend to have the master positioning range set automatically by means of the MotionFS_Ini function. This means that automatically the travel range of the motion master (if the motion master is not a JX2-CNT1) is set to the same positioning range. Then, the positioning range is set as has been illustrated in Fig. 1.

At automatic setting, the specified maximum cutting length MaxCuttingLength is used as a basis instead of the mechanical statuses of the motion master.

Example:

The motion-master-axis is a conveyor belt of 1,000 mm length. The cutting length is 533 mm.

If the positioning range of the motion master were to be set according to the drive mechanism, the following positioning limits would have to be specified in the motion master:

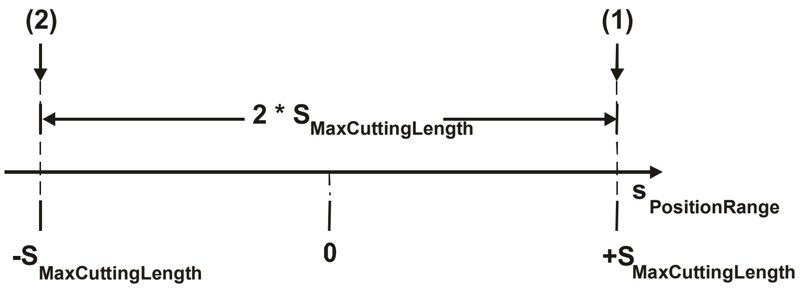
- 0 ... 1,000 mm

or

- -500 ... 500 mm

If the positioning range of the motion master is automatically set by means of the MotionFS_Ini function, the positioning limits in the motion master and in the sawing axis are set as follows after executing the function:

- -533 ... 533 mm




($s_{PositionRange}$) position range
($S_{MaxCuttingLength}$) maximum cutting length MaxCuttingLength
(1) Travel limit, positive (motion master) / maximum master position (sawing axis)
(2) Travel limit, negative (motion master) / minimum master position (sawing axis)

**Fig. 1 : Setting the positioning range of the master automatically**

Due to being set automatically, the master positioning range loses the absolute relation to the mechanism of the motion master. In a Flying Saw application, the absolute relation is not an imperative either, as the cuts are carried out in relation to one another, or else they are triggered by a hardware signal.

If, for certain reasons, the absolute relation to the drive mechanism is to be kept up, e.g. if between the sawing cycles absolute positiong runs are to be carried out by means of the motion master axis, it is up to the user to set the positioning range via the PLC program before carrying out the function MotionFs_Ini. The following must be given heed to:

A distinction between two cases must be made:

1st Case: The Motion Master is a JM-2xx:

In this case, the master positioning range in the sawing axis must be set to the same value as the negative and positive travel limit in the motion master itself, see chapter "Master-Slave Mode" in the document "jm2xx_at_jetcontrol_bi_xxxx_user_information.pdf".

Further, the greatest possible cutting length must fit at least twice into this positioning range. If this is not the case, the position range in motion master and sawing axis is set to an integer multiple of the mechanic position range, in order to meet the requirement again. In this case, overflows of the mechanic positioning range need to be counted by the user him/herself in the PLC program, lest the absolute relation get lost.

Example: The master-axis is a conveyor belt of 1,000 mm length. The greatest possible cutting length is 1,200 mm.

In this case, the user must set the positioning range in the motion master and the sawing axis as follows:

1. Calculating the minimum positioning range: 2 * greatest possible cutting length = 2,400 mm
2. Specify, how often the mechanical positioning range is to cover the minimum positioning range completely without splitting up the mechanical positioning range: 3 * 1,000 mm = 3,000 mm

Positioning range:

- 0 ... 3,000 mm

or

- -1,500 ... 1,500 mm

**Important**

**If the motion master is a JM-2xx, exactly the same positioning range must be defined for the master positioning range of the sawing axis, as has been defined for the travel range in the motion master.**

2nd Case: The Master is a JX2-CNT1:

In this case, there is no specification of the negative and positive travel limit in the motion master. The master positioning range can only be set in the sawing axis, see chapter "Master-Slave Mode" in the document "jm2xx_at_jetcontrol_bi_xxxx_user_information.pdf".

As in the first case, the greatest possible cutting length must at least fit twice into this positioning range. If this is not the case, here as well the positioning range to be set in the sawing axis must be an integer multiple of the mechanic positioning range of the motion master, in order to meet the requirements again. Please refer to the example above illustrating the first case.

## 2.3 Table Definition

For the acceleration and synchronous motion of the sawing axis, one table must be set up at least. The table is set up by means of the functions MotionFS_LoadTab_Linear and MotionFS_LoadTab_Sin2 of the function library, see 9.8.2 "MotionFS_LoadTab_Linear / MotionFS_LoadTab_Sin2“ on page 41.

Various tables of different acceleration values can be set up. During operation, changeovers between these tables can be made, see chapter 6.4 "Change of Tables" on page 24.

# 3 Production Period and Sawing Cycle

## 3.1 Production Period

A production period is defined as follows:

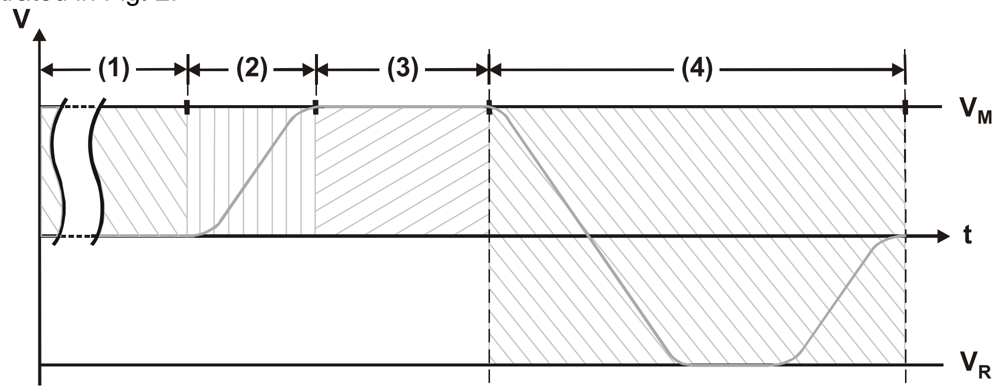
> A production period contains several, respectively a great number of sawing cycles. It will last until the presently active product format is terminated or changed or if an interrupt – e.g. due to an error – occurs.

At the beginning of a production period, initialisation by means of the function MotionFS_Reset must be made, see chapter 9.8.3 "MotionFS_Reset“ on page 45.

## 3.2 Sawing Cycle

A sawing cycle is defined as follows:

> One sawing cycle carries out one cut. In the JM-2xx, the sawing cycle consists of the phases illustrated in Fig. 2:




(1) 1. Phase: Waiting phase: Wait, until the master position has reached the automatically determined starting position
(2) 2. Phase: Synchronising phase / acceleration phase
(3) 3. Phase: Synchronous phase / sawing phase
(4) 4. Phase: Homeward voyage

**Fig. 2 Sawing Cycle**

The sawing axis is in the first to third phase of the table function, which is automatically started by a starting function (e.g. MotionFS_Start). For the fourth phase, the user has to end the table processing of the Flying Saw function via the PLC program, see chapter 3.6 "Disengaging“ on page 10.

Important

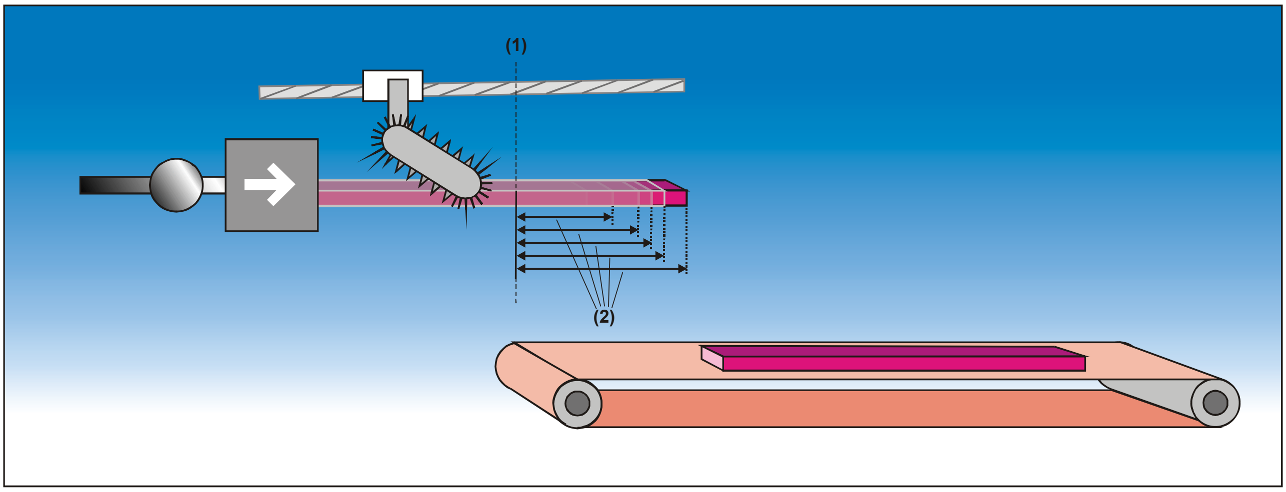
**The sawing axis must by no means reach the end of the table, as it would come to an abrupt halt there. Before reaching the end of the table, table processing must be terminated appropriately. The table length (length of the master positioning range of the table) is identical with the value of the parameter MaxCuttingLength, which is transferred at calling up the function MotionFS_Ini.**

## 3.3 Status of the Sawing Cycle

The function MotionFS_GetStartStatus is for querying the respective starting phase of the sawing axis (phase 1-3), see chapter 9.8.8 "MotionFS_GetStartStatus“ on page 53.

## 3.4 Displaying the Material Length

If a Flying Saw is applied, it might be desirable to display the already unwound material length before cutting.




(1) Synchronised position, related to sawing axis
(2) Material length at different instants

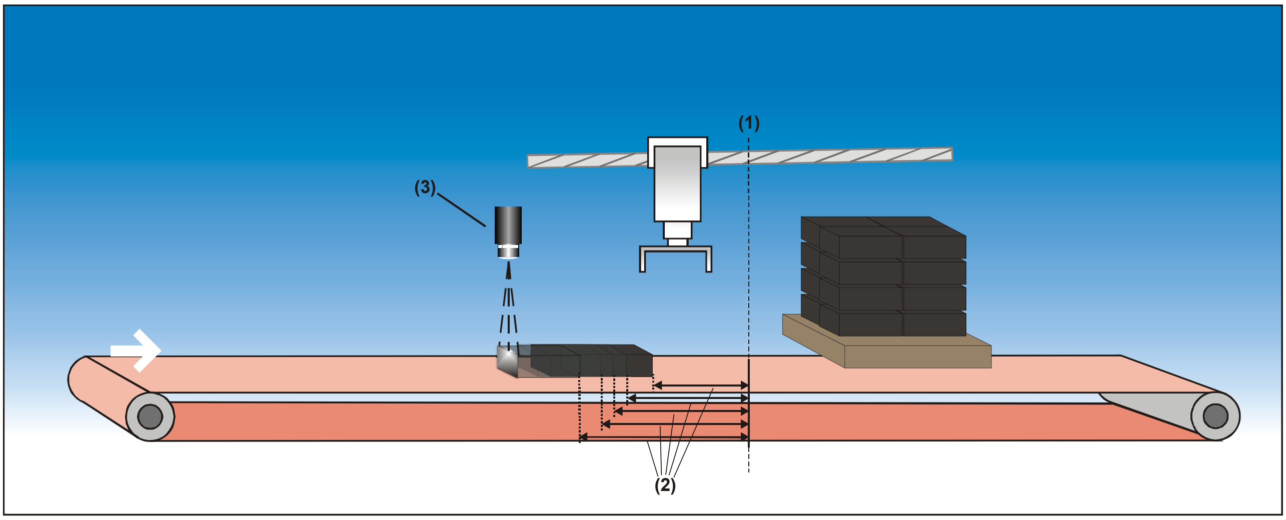
**Fig. 3 Determining the material length**

The function MotionFS_GetActCuttingLength reports back the material length at cycle start, see chapter 9.8.9 "MotionFS_GetActCuttingLength“ on page 55.

The material length can only be determined when classical Flying Saws are used.

## 3.5 Displaying the Distance between Synchronising Position and the Cutting Position

If a Flying Saw with hardware signal is applied, the distance between synchronising position and the cutting position can be determined. The function MotionFS_GetDistToCutPoint reports back the distance, see chapter 9.8.10 "MotionFS_GetDistToCutPoint“ on page 58.



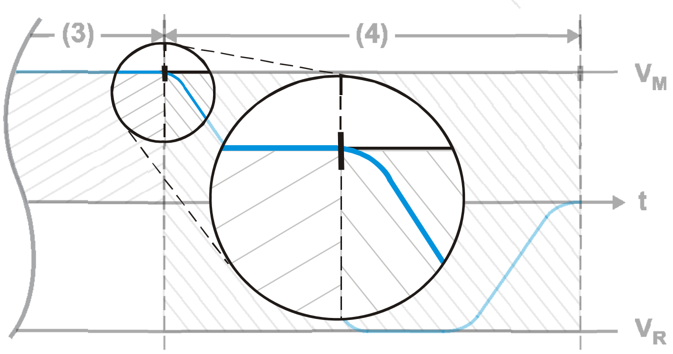

(1) Synchronised position, related to sawing axis
(2) Distance to the cutting position at various instances of time
(3) Material sensor

**Fig. 4 Determining the distance between synchronising position and the cutting position**

## 3.6 Disengaging

After the cut, the user must end table processing through sawing axis ("disengage the sawing axis"). Please refer to the following subchapter for further assistance.

### 3.6.1 Disengaging by immediate deceleration ramp:




(3) Synchronous phase
(4) Homeward voyage

**Fig. 5 Disengaging by immediate deceleration ramp:**

There are two ways of doing this:

1. Direct disengaging by homeward voyage positioning
2. Disengaging by deceleration ramp and a following homeward voyage positioning

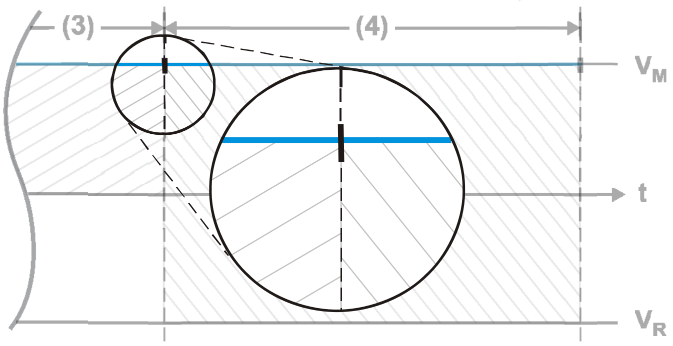
Example 1: Direct disengaging by homeward voyage positioning

```
...
//--- Homeward voyage: Start positioning, while
//    sawing axis is in table mode
//    ATTENTION: At returning, the acceleration value is used as deceleration value
       MotionMovePtp(Ax,all, absolute, 5000, 1500, <acceleration>, <deceleration>,
       <destination window>);
       WHEN MotionReadStatus(Ax, in destination window) CONTINUE;
       //--- As of this position, the sawing axis is in its starting position again
...
```

Example 2: Disengaging by deceleration ramp and a following homeward voyage positioning

```
 ...
 //--- Deceleration ramp with user-defined ramp, while
 //    the sawing axis is in table mode
     MotionStop(Ax, user-defined ramp, <deceleration>);
     //--- The bits "in the destination window" and "stopped" are in this case
     //    not supplied by JM-2xx. Instead, the bit "deceleration ramp" has to
     //    be queried.
     WHEN NOT MotionReadStatus(Ax, deceleration ramp) CONTINUE;
 //--- Homeward voyage positioning (see example 1)
     ...
```

### 3.6.2 Disengaging at constant speed:



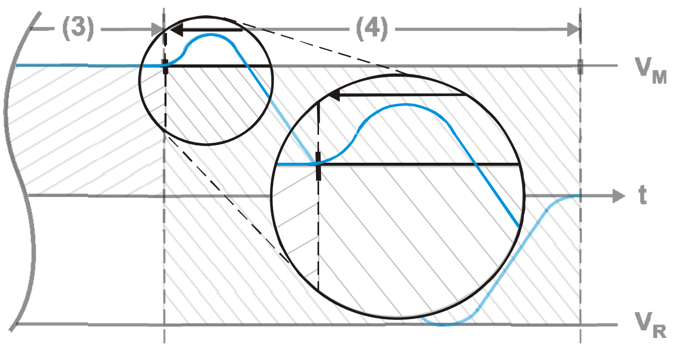

(3) Synchronous phase
(4) Homeward voyage

**Fig. 6 Disengaging at constant speed:**

Example

```
 ...
 //--- Decouple and keep up the speed, while
 //    the sawing axis is in table mode
     //--- Determine the actual speed of the sawing axis
         f_ActualSpeed:= MotionReadValue(Ax, ActualSpeed, MechanicalParts);
     //--- Determine the actual position of the sawing axis
         f_ActualPosition := MotionReadValue(Ax, ActualPosition);
     //--- Calculate a more distant position by means of the actual position value
         f_ActualPosition := f_ActualPosition + 5000;
     //--- Carry out positioning
         MotionMovePtp(Ax,All, Absolute, f_ActualPosition, f_ActualSpeed, *, *, *);
         WHEN MotionReadStatus(Ax, InDestinationWindow) CONTINUE;
     //--- Homeward voyage
         ...
```

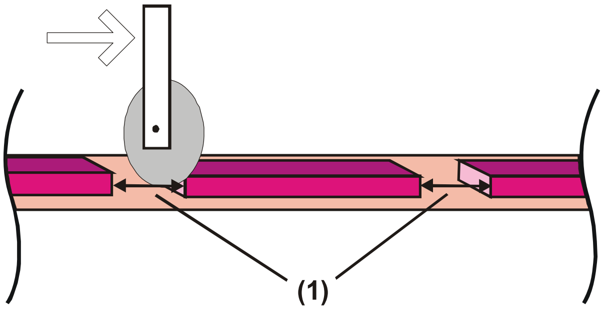
### 3.6.3 Disengaging with acceleration




(3) Synchronous phase
(4) Homeward voyage

**Fig. 7 Disengaging with acceleration**

Disengaging with acceleration can be helpful, for example, if the tools unit of the sawing axis is to make a gap between the parts that have been sawn apart.




(1) Gap between product parts

**Fig. 8 Making a gap**

Example:

```
...
//--- Decoupling during acceleration, while
//    sawing axis is in table mode
    //--- Determine actual speed of the sawing axis
        f_ActualSpeed:= MotionReadValue(Ax, ActualSpeed, MechanicalParts);
    //--- increase speed
        f_ActualSpeed:= f_ActualSpeed + 500;
    //--- Determine actual position of the sawing axis
        f_ActualPosition := MotionReadValue(Ax, ActualPosition);
    //--- Calculate a more distant position by means of the actual position value
        f_ActualPosition := f_ActualPosition + 5000;
    //--- Carry out positioning
        MotionMovePtp(Ax,All, Absolute, f_ActualPosition, f_ActualSpeed,
        <Acceleration>, *, *);
        WHEN MotionReadStatus(Ax, InDestinationWindow) CONTINUE;
    //--- Homeward voyage
        ...
```

## 3.7 Motion Stop

By the motion instruction MotionStop, the sawing axis can be stopped any time during the sawing cycle. In this case, it does not make any difference, whether the sawing axis is in table mode at that moment or not.

Example 1: with "Maximum Deceleration"

```
MotionStop(Ax, Maximum Deceleration);
//--- The bits "in the destination window" and "stopped" are in this case
//    not supplied by JM-2xx, when it is in table mode. Instead, the bit
"deceleration ramp" has to be queried.
WHEN NOT MotionReadStatus(Ax, DecelerationRamp) CONTINUE;
```

Example 2: with "User Defined Ramp"

```
MotionStop(Ax, user-defined ramp, <deceleration>);

//--- The bits "in the destination window" and "stopped" are in this case
//    not supplied by JM-2xx, when it is in table mode. Instead, the bit
"deceleration ramp" has to be queried.
WHEN NOT MotionReadStatus(Ax, DecelerationRamp) CONTINUE;
```

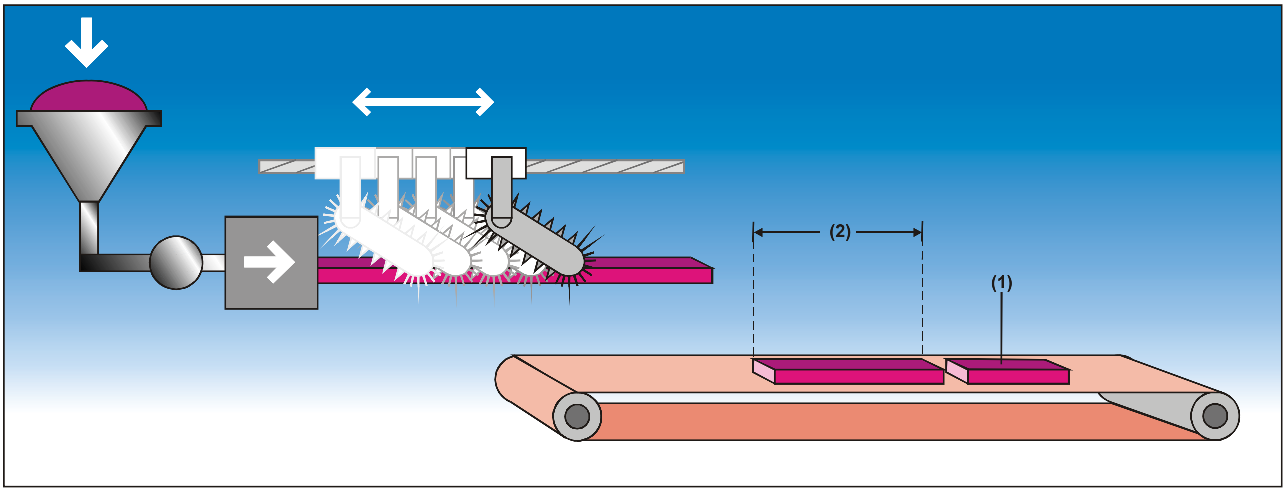
Important

**If the process is stopped, while a task is still waiting for synchronising in one of the starting functions (Motion FS_Start etc.), this task must be restarted. As, in the case of stopping the process, the synchronous status will not be reached, the starting function cannot stop by itself; for this, please refer to the example project in task t_Emerg.**

# 4 Classical Flying Saw

## 4.1 Variant 1: First Cut as Starting Position ("Head Cut")

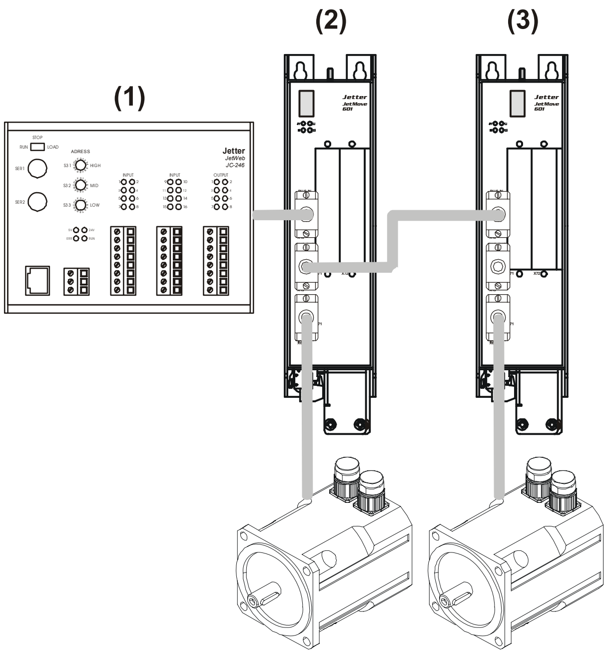
If the first cut is to set the starting position; it is called head cut. This cut is carried out without adjusting the tools unit to the material web, which means the material is cut at any position. Thus, the first piece that has been cut off (head cut) cannot be used. The following cuts are set in relation to this head cut and thus adjusted to the tools unit.



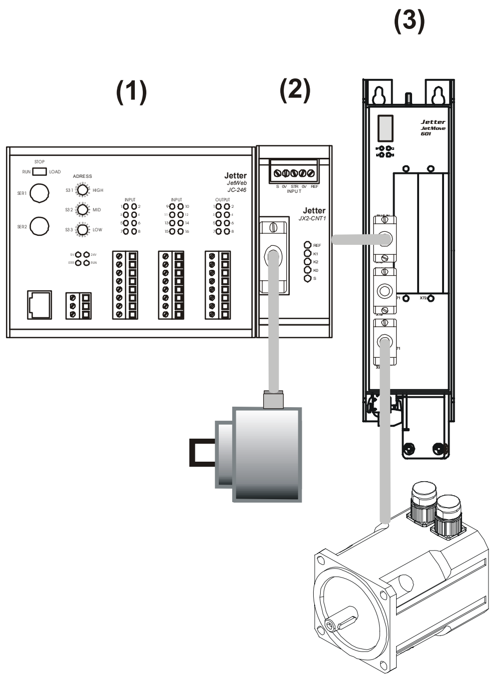

(1) Head cut
(2) Fixed cutting length

**Fig. 8: Classical flying saw with head cut**

### 4.1.1 Hardware configuration




(1) PLC
(2) Motion master: JM-2xx
(3) Sawing axis: JM-2xx




(1) PLC
(2) Motion master: JX2-CNT1
(3) Sawing axis: JM-2xx

**Fig. 9 Hardware configuration with JM-2xx**

**Fig. 10 Hardware configuration with JX2-CNT1**

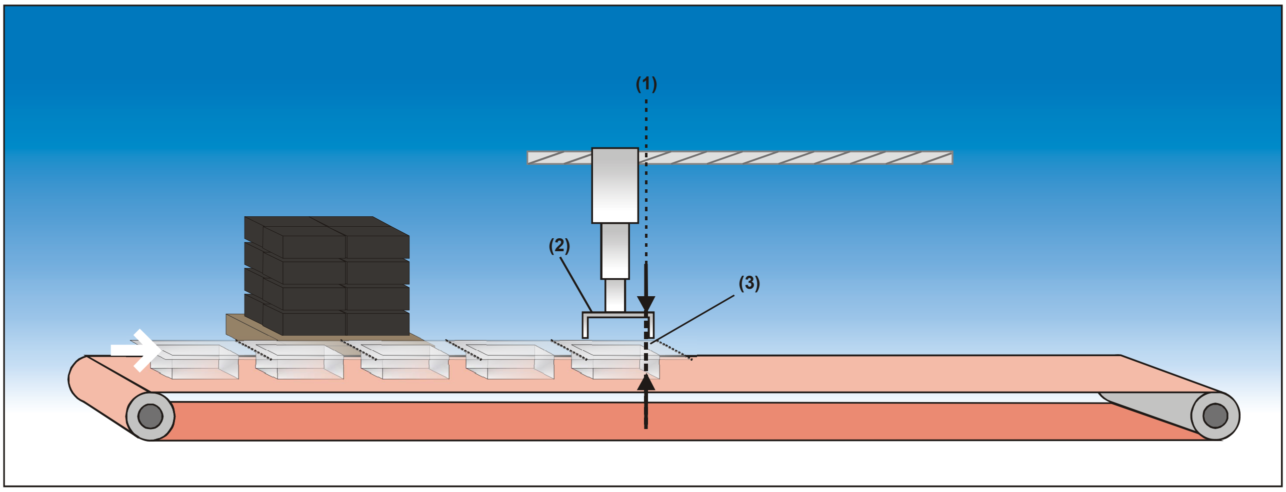
### 4.1.2 Sample program

See also the procedure "su_ClassicalHeadCut" in the sample project.

```
//--- Production period
      ...
      //--- Move the sawing axis to the starting position
          MotionMovePtp (Ax, All, Absolute, <SawingAxisStartPosition>,...)
          WHEN MotionReadStatus (Ax, InDestinationWindow) CONTINUE;

      //--- head cut
          MotionFS_StartInstant(&st_MotionFS_Slave0Data, &st_MotionFS_Slave0TabData0, 0,
                                &st_MotionFS_TmpFloatVars0);

      //--- carry out the cut
          ...

      //--- sawing cycle
          WHILE <cycle flag> DO       //---while cycle flag is set
              //--- Move the sawing axis to the starting position
                  MotionMovePtp (Ax, All, Absolute, <SawingAxisStartPosition>,...)
                  WHEN MotionReadStatus (Ax, InDestinationWindow) CONTINUE;

              //--- Start sawing axis
                  n_Return:= MotionFS_Start(&st_MotionFS_Slave0Data,
                                      &st_MotionFS_Slave0TabData0, &f_CuttingLength, 0,
                                      &st_MotionFS_TmpFloatVars0);

                  CASE n_Return OF    //--- Evaluation of the return value
                      c_FS_START_STAT_ERR_MASTER_POS_TOO_FAR:
                          DISPLAY_TEXT (<line>, <column>, 'ERROR: Master too far$');
                          b_AutoCycle:= FALSE;
                          RETURN;
                          BREAK;
                  END_CASE;

              //--- Carry out the cut
                  ...
          END_WHILE
          ...
```

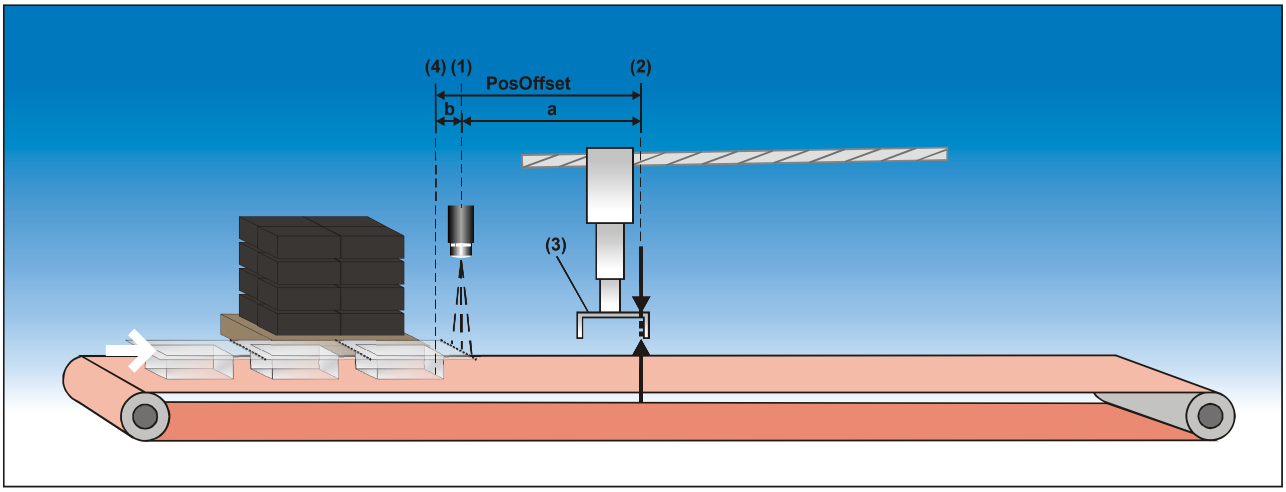
## 4.2 Variant 2: Starting by Adjusting Material and Tools Unit

Before carrying out the first step, material and tools unit are adjusted. This way, the cutting position on the material has been set exactly before carrying out the first cut. Adjustment can be done manually, e.g. by jog mode. With the help of a material sensor, it can also be carried out automatically.




(1) Starting position of the sawing axis
(2) Tools unit
(3) Adjusting position of the material is at the starting position of the sawing axis

**Fig. 11 Manual adjustment of material and tools unit**




(1) Responding position of the material sensor
(2) Starting position of the sawing axis
(3) Tools unit
(4) Adjustment point to the material
(a) Distance between material sensor and starting position of the sawing axis
(b) Distance between material sensing edge and orientation point at the material
(PosOffset) transfer parameters PosOffset for function MotionFS_StartPosCap

**Fig. 12 Automatic adjustment of material and tools unit**

### 4.2.1 General conditions

**Automatic adjustment:**

- The material sensor has to be positioned in appropriate distance PRECEDING the tools unit, see chapter 9.8.7 "MotionFS_StartCapPos“ on page 51.

- The distance between the responding position of the material sensor and the starting position of the sawing axis, PosOffset, must not be greater than the maximum positioning range of the motion master.

### 4.2.2 Hardware configuration

**Manual adjustment**

As shown above for variant 1.

**Automatic adjustment:**

As shown below for variant 1 of the Flying Saw with hardware signal

### 4.2.3 Sample program

**Manual adjustment**

```
VAR
     ...
     f_AlignPos:             FLOAT   AT %VL  65080;  //--- For storing the
                                                     //    adjustment position
     ...
VAR_END;

//--- production period
     ...
     //--- Set the position of the motion master to the adjustment position.
     //    Before this, the material and the tools unit have manually been adjusted at
     the starting position of the sawing axis
     MotionFS_GetMasterAlignPos(&st_MotionFS_Slave0Data,&st_MotionFS_Slave0TabData0,
     0,&f_AlignPos,&st_MotionFS_TmpFloatVars0);
     MotionHome ( Ax, SetBasicPosition, f_AlignPos);

     //--- Sawing cycle
     see above, at variant 1...
```

**Automatic adjustment:**

The sample application has been shown in Fig. 12. The distance between adjustment position at the material and the starting position of the sawing axis (a + b) is 90 mm + 500 mm = 590 mm. See also the procedure "su_ClassicalAutoAlign" in the sample project.

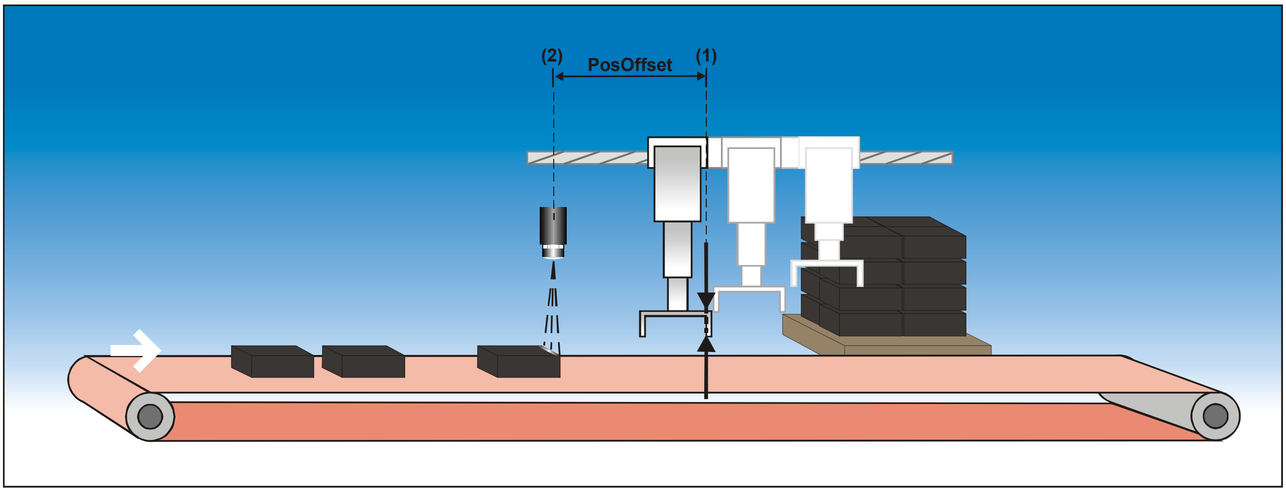
```
VAR
     ...
     f_CapturePos:           FLOAT   AT %VL  65081;  //--- Capture position
     ...
VAR_END;

 //--- Production period
     ...
     //  Move sawing axis to the starting position
         MotionMovePtp (Ax, All, Absolute, <SawingAxisStartingPosition>,...)
         WHEN MotionReadStatus (Ax, InDestinationWindow) CONTINUE;

     //--- Activate the capture function for the motion master position and wait,
     //    until capture event has set in;
     //    and save capture position in f_CapturePos
         ...(see sample project)
         f_CapturePos:= ...

     //--- Start first sawing cycle after the capture event
         MotionFS_StartCapPos(&st_MotionFS_Slave0Data,&st_MotionFS_Slave0TabData0,
                 &f_CapturePos,590,0,&st_MotionFS_TmpFloatVars0);

     //--- Carry out the cut
         ...
```

```
//--- Sawing cycle
    see above, at variant 1...
```

# 5 Flying Saw with Hardware Signal

## 5.1 Variant 1: Direct Start after Hardware Signal, Case 1

The sawing cycle is started directly after the corresponding hardware signal. The hardware signal synchronizes with the material in such a way, that between the hardware signal and the start of the corresponding sawing cycle, no further hardware signals are generated that might be able to start another sawing cycle.




(1) Starting position of the sawing axis
(2) Responding position of the material sensor
(PosOffset) transfer parameters PosOffset for function MotionFS_StartPosCap

**Fig. 13 Flying Saw with hardware signal – case 1**

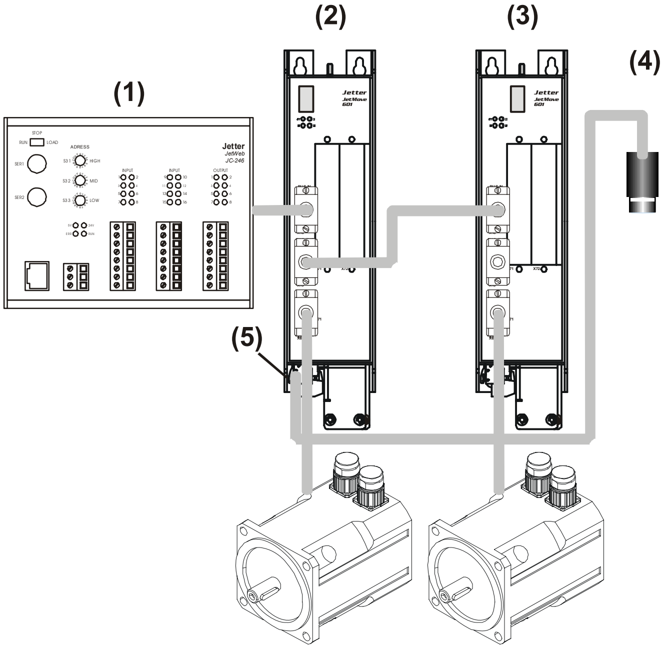
### 5.1.1 General conditions

- The material sensor has to be positioned in appropriate distance PRECEDING the tools unit, see chapter 9.8.7 "MotionFS_StartCapPos“ on page 51.
- Between one hardware signal and the next, the position of the motion master must not have covered the entire positioning range. This must be considered at setting the maximum and minimum position of the motion master in the sawing axis.
- The distance between the responding position of the material sensor and the starting position of the sawing axis, PosOffset, must not be greater than the maximum positioning range of the motion master.
- The time space between one hardware signal and the next must not be shorter than the duration of the sawing cycle.

### 5.1.2 Hardware configuration

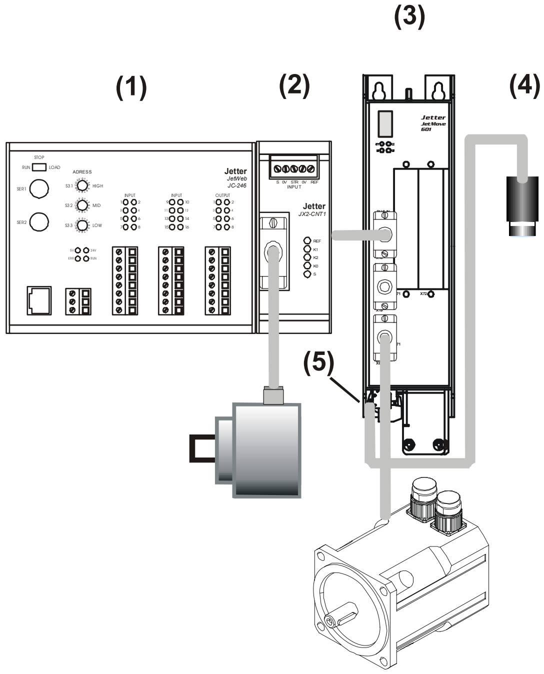
**When JX2-CNT1 is used as a motion master, the function "Referencing on the Fly" is applied by the sawing axis for capturing the master position.**

**Important**




(1) PLC
(2) Motion master: JM-2xx
(3) Sawing axis: JM-2xx
(4) Material sensor
(5) Input "INPUT" at the motion master

**Fig. 14 Hardware configuration with JM-2xx**




(1) PLC
(2) Motion master: JX2-CNT1
(3) Sawing axis: JM-2xx
(4) Material sensor
(5) Input "INPUT" at the sawing axis

**Fig. 15 Hardware configuration with JX2-CNT1**

### 5.1.3 Sample program

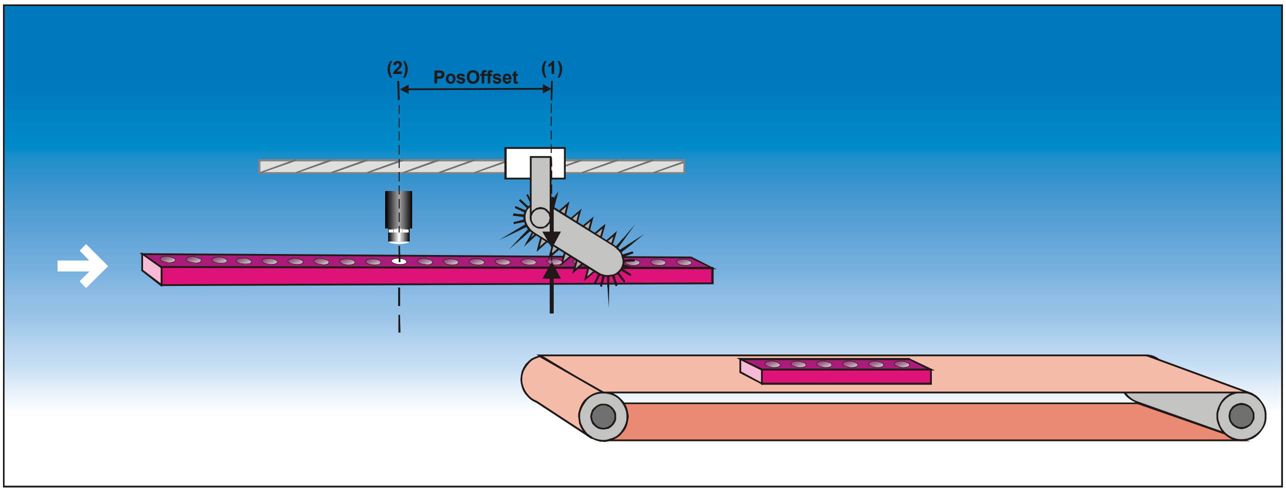
The sample application has been shown in Fig. 13. The distance between responding position of the material sensor and the starting position of the sawing axis (PosOffset) is 1000 mm. Also refer to the procedure "su_DirectTriggerEndlessCutCase1" in the sample program.

```
//--- Production period
     ...
     //--- Sawing cycle
         WHILE <cycle flag> DO       //--- While cycle flag is set...
     //  Move sawing axis to the starting position
         MotionMovePtp (Ax, All, Absolute, <SawingAxisStartingPosition>,...)
         WHEN MotionReadStatus (Ax, InDestinationWindow) CONTINUE;

     //--- Activate the capture function for the motion master position and wait,
     //    until capture event has set in;
     //    and save capture position in f_CapturePos
         ...(see sample program)
         f_CapturePos:= ...
```

```
//--- Start first sawing cycle after the capture event
               MotionFS_StartCapPos(&st_MotionFS_Slave0Data,
                              &st_MotionFS_Slave0TabData0,&f_CapturePos,1000,0,
                              &st_MotionFS_TmpFloatVars0);

//--- Carry out the cut
               ...

     END_WHILE
     ...
```

## 5.2 Variant 1: Direct Start after Hardware Signal, Case 2

In case 2, the hardware signal is counted additionally; not before a certain number of signals have been given, the sawing cycle is started.




(1) Starting position of the sawing axis
(2) Responding position of the material sensor
(PosOffset) transfer parameters PosOffset for function MotionFS_StartPosCap

**Fig. 16 Flying Saw with hardware signal – case 2**

### 5.2.1 General conditions

General conditions as listed for case 1 above, plus additionally:

- Counting the hardware signals and starting the sawing axis must be carried out in separate tasks, in order to have them processed simultaneously.
- The time intervals between the hardware signals must be long enough for the counting task still to recognise each signal individually. In this case, the task cycle time of the controller must be considered.

### 5.2.2 Hardware configuration

As shown above for case 1.

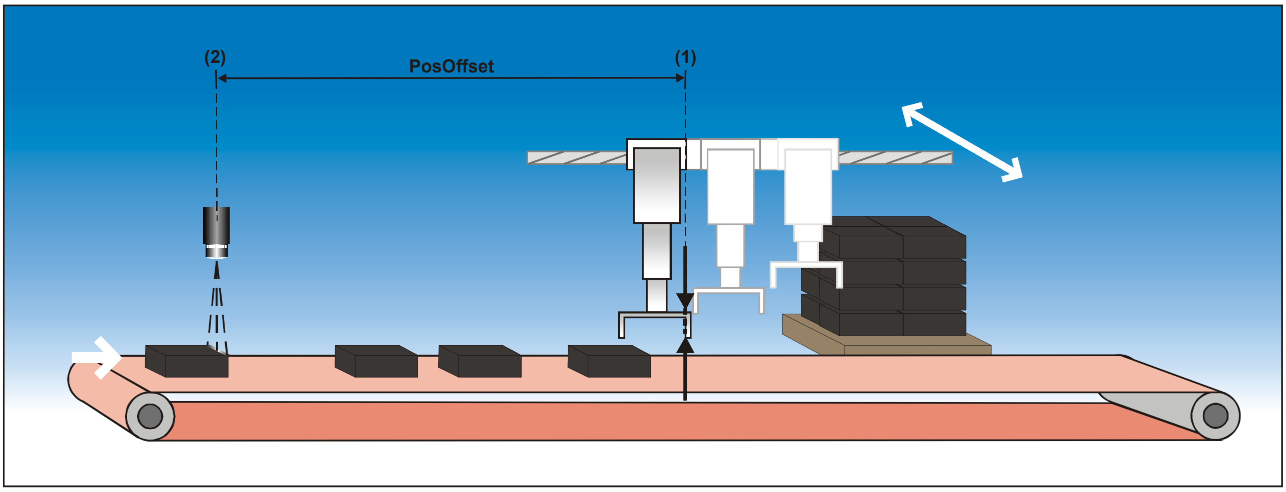
### 5.2.3 Sample program

As in the example for case 1, additional counting of capture events is to be done in a separate task. Also refer to the procedure "su_DirectTriggerEndlessCutCase2" in the sample program.

```
TASK t_Counting
     ...
     //--- Counting the capture events
         IF <CountingActiveFlag> THEN
             //--- Activate and evaluate capture function
                 ...(see sample program)
             //--- Increase counter of capture events
                 ...
             //--- When the capture event counter has reached the maximum value, save the
             //    position value
                 f_CapturePos:= ...

         END_IF;

     ...
END_TASK;

TASK Production Period
     ...
     //--- Production period
             ...
         //--- Sawing cycle
             WHILE <cycle flag> DO       //--- While cycle flag is set
                 //--- Move sawing axis to the starting position
                     MotionMovePtp (Ax, All, Absolute, <Sawing axis starting
                     position>,...)
                     WHEN MotionReadStatus (Ax, In DestinationWindow) CONTINUE;

                 //--- Wait for the required number of capture events
                 //    to have occurred, in this example = 7
                     WHEN <CaptureCnt>= 7 CONTINUE;
                         //--- Re-initialise the capture unit for the next
                         //    counting
                             ...

                 //--- Start after the capture event
                 //    Variable f_CapturePos has been described in task t_Counting
                     MotionFS_StartCapPos(&st_MotionFS_Slave0Data,
                                 &st_MotionFS_Slave0TabData0,&f_CapturePos,1000,0,
                                 &st_MotionFS_TmpFloatVars0);

                 //--- Carry out the cut
                     ...

             END_WHILE
             ...

     ...
END_TASK;
```

## 5.3 Variant 2: Accumulating of Several Hardware Signals

Before the sawing cycle corresponding to a certain hardware signal is started, a certain number of further hardware signals accumulate which start a sawing cycle as well.




(1) Starting position of the sawing axis
(2) Responding position of the material sensor
(PosOffset) transfer parameters PosOffset for function MotionFS_StartPosCap

**Fig. 17 Flying saw with accumulating hardware signals**

### 5.3.1 General conditions

General conditions as listed for variant 1, case 1 above, and additionally:

- A FIFO for saving the motion master position at receiving the hardware signal must be used.
- Filling the FIFO and starting the sawing axis must be carried out in separate tasks, in order to have them processed simultaneously.

### 5.3.2 Hardware configuration

See above for variant 1.

### 5.3.3 Sample program

In the example, the parameter PosOffset is 10,000 mm. Also refer to the procedure "su_FifoTriggerEndlessCut" in the sample program.

```
TASK t_FIFO
    ...
    //--- Filling the FIFO
        IF <FIFO_FillActiveFlag> THEN
            //--- activate and evaluate capture function
                ...(see sample program)
            //--- At a capture event, fill the next FIFO element with the FIFO input
            //    pointer value
                ...(see sample program)
            //--- increase FIFO input pointer value
                ...(see sample program)
        END_IF;

    ...
END_TASK;

TASK production period
    ...
```

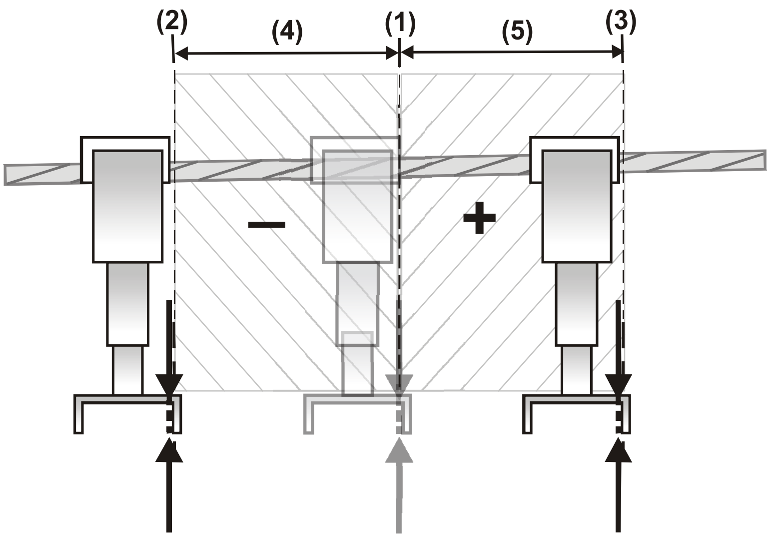
```
     //--- Production period
             ...
         //--- Sawing cycle
             WHILE <cycle flag> DO       //--- While cycle flags are set
                 //--- Move the sawing axis to the starting position
                     MotionMovePtp (Ax, All, Absolute, <SawnigAxisStartingPosition>,...)
                     WHEN MotionReadStatus (Ax, In DestinationWindow) CONTINUE;

                 //--- Wait for the next FIFO element to have got an entry
                     WHEN <FIFO_OutputPointer> <> <ComparisonValue> CONTINUE;

                 //--- set CapturePositionVariable
                     f_CapturePos:= @FIFO_OutputPointer;
                     ...

                 //--- Increase value of FIFO output pointer
                     ...(see sample program)

                 //--- Start after the capture event
                     MotionFS_StartCapPos(&st_MotionFS_Slave0Data,
                                 &st_MotionFS_Slave0TabData0,&f_CapturePos,10000,0,
                                 &st_MotionFS_TmpFloatVars0);

                 //--- Carry out the step
                     ...

             END_WHILE
             ...

     ...
 END_TASK;
```

# 6 Further Required Features

## 6.1 Change the Starting Position of the Sawing Axis

If the starting position of the sawing axis is not the same as has been specified at carrying out the function MotionFS_LoadTabLinear, respectively MotionFS_LoadTabSin2, the cutting position has been shifted by the difference between set and actual starting position. This means, that in case of a classical Flying Saw, this difference must still be added to the cutting length, while in case of a Flying Saw with hardware signal, this difference must be added to the distance (PosOffset) between material sensor and starting position.

A change of the set starting position is necessary at cuts without return, for example; please refer to the following chapter.



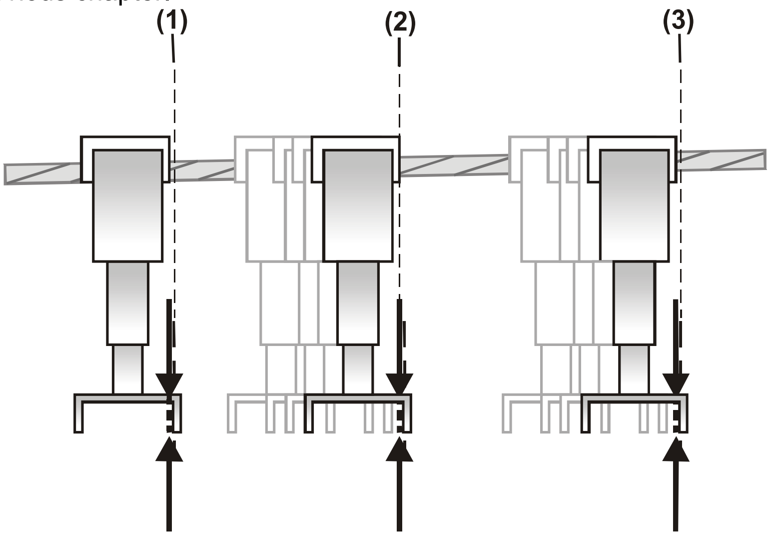
(1) Starting position of the sawing axis as specified in the table definition
(2) Starting position with negative difference (4) to the specified starting position
(3) Starting position with positive difference (5) to the specified starting position

**Fig. 18 Changing the starting position**

## 6.2 Cut Without Homeward Voyage

In order to achieve shorter cutting lengths than are permitted by the cycle time including homeward voyages, some cuts can be made without a homeward voyage. This way, the time for homeward voyage is saved. The number of those cuts is determined by the travel of the traversing axis.

In this case, the length of the cut to be made next must be corrected after each cut performed without homeward voyage: The difference between actual and set starting position must be added to the required cutting length, see previous chapter.




(1) Starting position cycle n (set starting position)
(2) Starting position cycle n+1
(3) Starting position cycle n+2

**Fig. 19 Cuts without homeward voyage**

## 6.3 Speed Difference between Sawing Axis and Motion Master

This applies to the following examples:

- Motion master and sawing axes have got different positioning units: The motion master, for example, is a conveyor belt (position in mm), while the sawing axis is a cylinder (position in °).
- Motion master and sawing axes are rotatory axes of varying circumferences.
- Diagonal cutter: Both sawing axis and motion master are linear axes, yet they do NOT run in parallel mode to each other.

In these cases, the parameter TabGearRatio is applied. This parameter is specified at carrying out the function MotionFS_LoadTab_Linear / MotionFS_LoadTab_Sin2. Normally, the parameter equals 1. In these applications, it is <> 1.

This parameter specifies the ratio, by which the sawing axis moves slower or faster than the motion master during the synchronous phase.

TabGearRatio < 1 means that the sawing axis moves slower than by TabGearRatio = 1
TabGearRatio > 1 means that the sawing axis moves faster than by TabGearRatio = 1

### 6.4 Change of Tables

If the parameters of a cut are to change in short instances, the table can be changed from sawing cycle to sawing cycle. A change of parameters would be, for example, a significant change of the motion master speed. Another significant change would be the acceleration of the sawing axis, in order to carry out a shorter cut, for example.

A prerequisite for the change of tables is an already existing definition of the table that is to be changed to.

In case of the application "Flying Saw with Hardware Signal", changing tables is very easy. At each calling up the starting function MotionFS_StartCapPos, the respective table is specified.

In case of a classical Flying Saw application, some more details are to be considered: By calling up the function MotionFS_StartChangeTab, the next sawing cycle processing the newly selected table is started. For the cycle after the changeover, the function MotionFS_Start can be called up again; it will process the newly selected table. In order to return to the former table, the process is once more started by MotionFS_StartChangeTab. It is also possible to change at every cut; this means the cycle must always be started by the function MotionFS_StartChangeTab.

**Example**

Please also refer to chapter 9.8.5 "MotionFS_StartChangeTab“ on page 48.

```
...
//--- Change from table 0 to table 1
    MotionFS_StartChangeTab(&st_MotionFS_Slave0Data,&st_MotionFS_Slave0TabData1,
                            &st_MotionFS_Slave0TabData0,...);
...

...
//--- Further cuts by table 1
    MotionFS_Start(&st_MotionFS_Slave0Data,&st_MotionFS_Slave0TabData1,
                                &f_CuttingLength,0,&st_MotionFS_TmpFloatVars0);
...

...
//--- Change back from table 1 to table 0
    MotionFS_StartChangeTab(&st_MotionFS_Slave0Data,&st_MotionFS_Slave0TabData0,
                            &st_MotionFS_Slave0TabData1,...);
...

...
//--- Further cuts by table 0
    MotionFS_Start(&st_MotionFS_Slave0Data,&st_MotionFS_Slave0TabData0,
                                &f_CuttingLength,0,&st_MotionFS_TmpFloatVars0);
...
```

## 7 Cut Accuracy

### 7.1 Inaccuracy of the Transmission Specifications

The cutting length, respectively the distance to the material sensor is specified in millimeters, respectively degrees. If the specifications for data transmission from motor encoder to the drive mechanism (in millimeters, respectively degrees) are not precise, the cuts will not be precise, either. Checking the cut products will always prove them to be either too short, respectively too long.

If the transmission ratio has not been specified precisely, one millimeter, respectively degree specified at the drive mechanics is actually slightly less or more than a millimeter or degree. Cutting can only be as precise as is the specification in millimeters or degrees.

This applies to the specifications of data transmission between motor encoder and the drive mechanism of the motion master and the sawing axis. Using a JX2-CNT1 also influences the transmission ratio from incremental resolution to millimeters, respectively degrees. The settings for this are carried out within the sawing axis parameters.

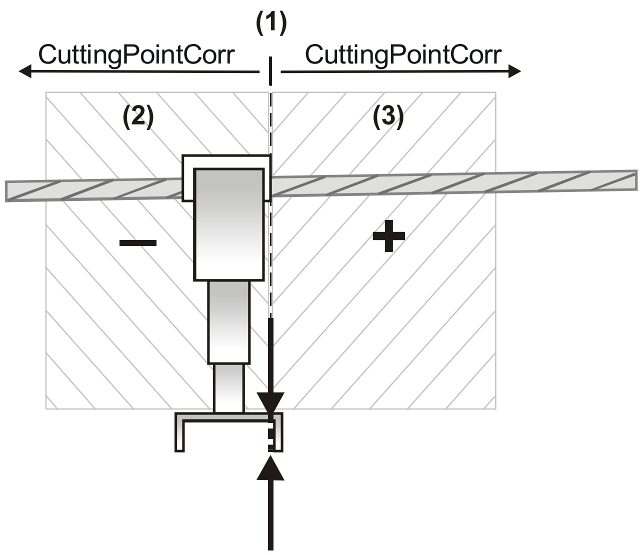
## 7.2 Correction of the cutting position

The cutting position can be corrected manually by the cutting position correction function during sawing cycles. For this, the parameter CuttingPointCorr is used in the structure MOTION_FS_SLAVE_DATA; please refer to chapter 9.5.1 "„MOTION_FS_SLAVE_DATA“ on page 33. By means of this parameter, a correction offset is defined, which is activated at starting the next cycle. After this, the parameter is reset to 0.

**Example**

In the example, the correction offset is set to 10 mm at activating a visualisation key. It can also be entered into the CuttingPointCorr parameter via a setup screen.

```
TASK <e.g. key management of the visu>
      ...
      //--- Shifting back the cutting position by 10 mm (the cuts become longer)
          st_MotionFS_Slave0Data.CuttingPointCorr:= 10;
      ...
```




(1) Actual cutting position
(2) Negative value range for CuttingPointCorr: Shift of the cutting position in moving direction of the master: Cutting length is increased
(3) Positive value range for CuttingPointCorr: Shift of the cutting position opposite the moving direction of the master: Cutting length is decreased

**Fig. 20 Correction of the cutting position**

The cutting position correction by means of the CuttingPointCorr is only possible with the classical Flying Saw, this means, at starting the sawing cycle by the functions MotionFS_Start and MotionFS_StartChangeTab. In the case of Flying Saws with hardware signal, i.e. at starting by the function

MotionFS_StartCapPos, the transfer parameter PosOffset must be corrected for a correction of the cutting position; for this, please turn to chapter 9.8.7 "„MotionFS_StartCapPos“ on page 51.

The correction offset is also dependent on the specifications of transmission between motor and drive mechanics. If the transmission specifications are not precise, the correction offset at the drive mechanics is slightly greater or smaller than the set value in mm respectively °. If, at the correction, the cutting position is not shifted exactly by the correction offset on the material, the transmission ratio of the master is not correct.

## 7.3 Adjustment of Synchronous Operation

If inadvertently the sawing axis is moving faster or slower than the motion master during the synchronous phase, this indicates imprecise specification of the transmission ratio between motor and drive mechanics. The transmission ratio has not been set precisely either at the motion master, or at the sawing axis, or at both of them.

This inaccuracy can manually be determined by changing the synchronous factor. The synchronous factor is defined by the structure MOTION_FS_SLAVE_DATA via the parameter SyncFactor; for this, please refer to chapter 9.5.1 "MOTION_FS_SLAVE_DATA“ on page 33 and chapter 9.8.12 "MotionFS_SetSyncCorrection“ on page 62. Normally, the synchronous factor is 1. By changing the factor, the sawing axis becomes either slower or faster than the motion master. The following values are to illustrate this:

| | |
|---|---|
| SyncFactor = 0.5: | The sawing axis is half as fast as the motion master |
| SyncFactor = 1: | The sawing axis is exactly as fast as the motion master |
| SyncFactor = 2: | The sawing axis is double as fast as the motion master |

Changing the synchronous factor does not become effective before the next sawing cycle.

**Important** **Adjustment of the synchronous operation during setup serves for determining inaccuracies of the transmission ratios. After the setup phase, adjustment of the synchronous operation should be made by means of the transmission ratios, in order to allow the synchronous factor for normal operation to be set to 1 again.**

**Important** **At each change of the synchronous factor, the cutting position automatically shifts as well. This shift can be corrected manually by the function of cutting position correction.**

**Example**

In the example, the synchronous factor is changed by 0.01 at activating a visualisation key by means of the function MotionFS_SetSyncCorrection. Yet, the synchronous factor can also be set via a setup screen in the parameter SyncFactor directly.

```
TASK <e.g. key management of the visu>
      ...
      //--- Increment synchronous factor by 0.01, i.e. sawing axis will travel faster
      //    than before.
          f_SyncFactorCorr:= 0.01;
          MotionFS_SetSyncCorrection(&st_MotionFS_Slave0Data,&f_SyncFactorCorr,0);
      ...
```

The synchronizing factor described here has got the same effect as the TabGearRatio parameter specified in the table definition, see chapter 9.8.2 "MotionFS_LoadTab_Linear / MotionFS_LoadTab_Sin2“ on page 41. What is special in this case is that at applying TabGearRatio, the synchronizing factor has already been integrated in the table profile. In this case, TabGearRatio is only applied to the synchronous motion of the table profile.

By means of the parameter SyncFactor, the synchronous factor can be changed dynamically. Yet, the synchronous factor of a value <> 1 has an effect on the specified acceleration as well. This means the sawing axis does not accelerate by the ratio which has been specified in the table definition, if a synchronous factor of <> 1 has been set. The acceleration ratio is either greater or smaller, dependent on the value of the synchronous factor.

## 7.4 Dead Time Compensation at High Master Speed

The sawing axis does not adjust to the master position at exactly the instance at which it is recognised by the motion master. Both the communication time between motion master and sawing axis and the time interval of the sawing axis adjusting to the master position contribute to a dead time. Due to this dead time, the sawing axis always follows the motion master by a short distance. The greater the master speed, the greater is the absolute distance between sawing axis and motion master.

This distance can lead to cut inaccuracies, if the master speed is both high **and** not constant, yet continually varying to a great extent. By means of a dead time compensation, the actual position of the sawing axis is corrected by the dead time value. This way, cut accuracy is improved.

By default, dead time compensation is not active. For setting and activating the dead time, please refer to the document "jm2xx_at_jetcontrol_bi_xxxx_user_information.pdf".

# 8 Troubleshooting

## The sawing axis does not start off

**Possibility 1: The applied table has not been defined**

Solution: Define a table

**Possibility 2: The motion master has already moved too far**

Soolution at the start by the functions MotionFS_Start and MotionFS_StartChangeTab:
Reduce the speed of the motion master.

Solution at the start by the function MotionFS_StartInstant:
Increase c_FS_STARTINSTANT_TIME_OF_PROCESS.

Solution at the start by the function MotionFS_StartCapPos:
Increase the mechanical distance between material sensor and starting position.

## Cutting position or cutting length is imprecise

Please refer to chapter 7 "Cut Accuracy“ on page 24.

## The cutting position correction is imprecise

Please refer to chapter 7.2 "Correction of the cutting position“ on page 25.

## The sawing axis is not synchronous with the motion master, which means that during the cut, the tool presses against the side of the material

Please refer to chapter 7.3 "Adjustment of Synchronous Operation“ on page 26.

## The sawing axis travels too far

**Possibility: The travel range of the sawing axis is too small for the set synchronous factor**

Solution: Enlarge the travel range

# 9 Function Library "Flying Saw"

## 9.1 What is the Function Library "Flying Saw"?

The function library is a collection of tested functions that are to support the Jetter programmer of JetControl PLCs in programming Flying Saw applications by JetSym ST.

The library contains functions and the corresponding definitions of data structures and constants. By means of te libraries, the programmer can implement and test the application program faster. This means programming times decrease.

## 9.2 Please observe the following:

1. Incorporating a function library increases the program size of the JetSym ST project.

## 9.3 Functions in JetSym ST

The following parameters have been set in the programming language JetSym ST for programming functions:

1. Transfer parameters are always of the INT type
2. Feedback parameters are always of the INT type

If as transfer parameters of the FLOAT type, or if entire structures have got to be transferred to a function, the register address of the respective Float variable, respectively the start address of the structure are specified. The addresses are of the INT type.

### Example

If the function xyz(*p_FloatPointer*, *p_StrukturPointer*) is applied, the Float variable *f_IAmFloat* and the structure *st_IAmStructure* are transferred.

**Function call-up using the & operator**

```
xyz(&f_IAmFloat, &st_IAmStructure);
```

**Function call by absolute addresses**

```
xyz(65051, 450);
```

**Processing within the function**

Within the function, access to the transfer parameters is made via indirect address:

```
...
VAR
     pl_HelpStructurePointer:    INT;       //--- Declaration of a
                                            //    local help pointer,
                                            //    in order for the
                                            //    variable containing
                                            //    the startng address
                                            //    not to have to be
                                            //    changed
END_VAR;
...
//--- Processing the float variables
..... @p_FloatPointer * 1.23;               //--- float variable
                                            //    is not used
                                            //    within the formula
...
//--- Processing the structure
pl_HelpStructurePointer:=p_StructurePointer +1;  //--- Set HelpPointer
```

```
                                           //    to 2nd element of
IF @pl_HelpStructurePointer > 100 THEN    //--- the structure
     ...                                   //    within an IF
                                           //    condition
...
```

## 9.4 Incorporating the Library Files

The function library consists of two files:

- Header file
- Function file

In order to use the library, both files must be incorporated in the JetSym ST project.

Incorporating is carried out by specifying the file names together with the compiler directive #INCLUDE in the main program file. In this case, specification must be made before the first TASK statement:

**Main program file:**

```
//--- Definition of the types and constants
  ...
#INCLUDE "LibJM2xxFS_h.stp"           // Including the header file
  ...
//--- Definition of the functions
  ...
#INCLUDE "LibJM2xxFS.stp"             // Including the function file
  ...
//--- Main program
  ...
TASK 0                                // The first task starts here
```

If the files are not in the same directory as the JetSym ST project to which they are to be stored, the entire path containing the complete file name must be specified in the INCLUDE instruction. Or else, the files must be included in the project tree under "Program" and "Include", see Fig. 21. Then, the entire path is known by means of including the file.

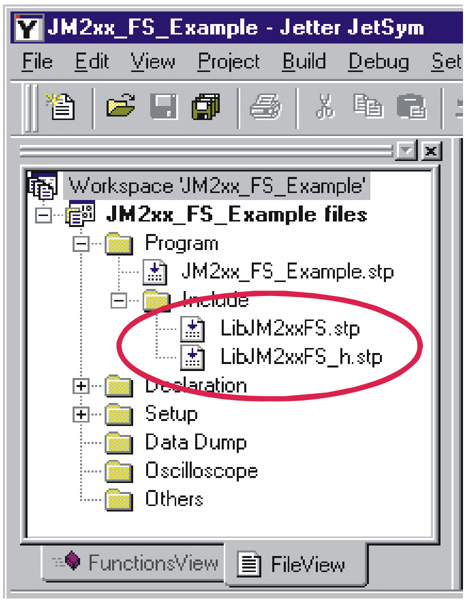



**Fig. 21 Include-files in the project tree**

## 9.5 Types

| Survey of Types |
|---|
| **Structures** |
| **MOTION_FS_SLAVE_DATA** |
| Definition of data for a Flying Saw axis. |
| **MOTION_FS_TAB_DATA** |
| Definition of data to be compiled in a table in which the acceleration and synchronous motion values of a Flying Saw are set. |
| **MOTION_FS_TMP_FLOATVARS** |
| Definition of the temporary FLOAT variables which are used as FLOAT buffers for calculations of Flying Saw functions. |

The structure data types listed above are used by the functions of the function library. In order for the functions to make use of the structures, at least one variable has to be declared for every structure in the program range VAR and END_VAR. For each variable, a register address must be defined then, at which the structure is to start. All variables of the structures **must** be completely located in the FLOAT register area of the controller.

For each available Flying Saw axis (several sawing axes can be part of one system bus string), one variable of the structure MOTION_FS_SLAVE_DATA must be created.

For each available sawing axis, at least one table must be defined, by which the acceleration and synchronous motion for the sawing cycle is set. For a Flying saw axis, several tables can be applied; the user can change between them. This means that for each table of a sawing axis, at least one variable of the structure MOTION_FS_TAB_DATA must be created.

At calling up some Flying Saw functions, a variable of the structure MOTION_FS_TMP_FLOATVARS must be added as a parameter. The structure is internally used for storing intermediate values for calculations etc. As long as the Flying Saw functions are called up just one after the other, the declaration of only one variable of this structure is sufficient. In this case, it does not matter, for which Flying Saw axis or for which table the function is executed. As soon as the functions are carried out in parallel mode, this means that several tasks are carried out simultaneously, a corresponding number of variables must be available in this structure. To each function, which is carried out simultaneously with another one, an individual variable of this structure is committed.

Normally, the structures are initialised, modified, and read by the Flying Saw functions. The controller program only needs to declare and make available the structure variables. In general, the controller program should not change the structure parameters outside the Flying Saw functions. An exception are the parameters CuttingPointCorr and SyncFactor of the structure MOTION_FS_SLAVE_DATA. These parameters have to be described on the user side of the controller program, if a correction of the cutting position or synchronising are to be carried out; please refer to chapter 7 "Cut Accuracy“ on page 24. Yet, the structure files can be read via the controller program or monitored on the JetSym setup pages.

Some of the data that are stored within the structures, e.g. MOTION_FS_TAB_DATA, are also available as parameters of the respective sawing axis. For optimising reasons, and for keeping bus accesses to a minimum and thus saving runtime, these data kept in structures have additionally been stored to the controller memory.

**Example**

In the following example, all variables required for a Flying Saw axis, are declared in a JC-24x by a table in the FLOAT register array.

```
VAR
     ...
     st_MotionFS_Slave0Data:     MOTION_FS_SLAVE_DATA AT %VL 65024;

     st_MotionFS_Slave0TabData0: MOTION_FS_TAB_DATA AT %VL 65024 +
                                 SIZEOF(st_MotionFS_Slave0Data);

     st_MotionFS_TmpFloatVars:   MOTION_FS_TMP_FLOATVARS AT %VL 65024 +
                                 SIZEOF(st_MotionFS_Slave0Data) +
                                 SIZEOF(st_MotionFS_Slave0TabData0);

     ...

 VAR_END;
```

### 9.5.1 MOTION_FS_SLAVE_DATA

**Definition**

```
MOTION_FS_SLAVE_DATA:
	STRUCT
		SlaveBase, MasterPosMax, MasterPosMin, MasterPosRange, LastTabOffsM, ActTabOffsM,
		ActMasterSyncPos, ActSyncPosTabIdx, MaxCuttingLength, CuttingPointCorr, SyncFactor,
		Reserved0: FLOAT;
	END_STRUCT;
```

**Parameters**

SlaveBase: Start address of the JM-2xx at the system bus of the Flying Saw axis is, for example, 12000; this represents the first JM-2xx directly following the controller JC-24x.

MasterPosMax: The maximum master position in mm, respectively °.

MasterPosMin: The minimum master position in mm, respectively °.

MasterPosRange: The master position range in mm, respectively °, dependent on MasterPosMax and MasterPosMin.

LastTabOffsM: Master offset of tables in mm, respectively ° of the latest sawing cycle

ActTabOffsM: Master offset of tables in mm, respectively ° of the present sawing cycle

ActMasterSyncPos: Synchronous position of the master in mm, respectively ° of the present sawing cycle The synchronous position is the position, as of which the Flying Saw axis is synchronous with the master axis.

ActSyncPosTabIdx: Index of the table element containing the synchronous positions of the actual sawing cycle.

MaxCuttingLength: Greatest possible cutting length in mm, respectively °. Here, the value of the greatest possible cutting length MaxCuttingLength has been stored, which is commited to the function MotionFS_Ini at calling it up.

CuttingPointCorr: Correction value of the cutting position. Here, a relative correction value must be entered in mm, respectively °, if a correction of the cutting position is to be carried out. At the next sawing cycle, the correction will be activated. After the correction, the parameter is reset to zero automatically. Please refer to chapter 7.2 "Correction of the cutting position“ on page 25.

Default value: 0
Value range: FLOAT limits,
negative values = increase cutting length
positive Werte = decrease cutting length

SyncFactor: Synchronising factor Here, a synchronising factor <> 1 must

be specified, if synchronisation between motion master and motion slave (=sawing axis) is to be corrected.
The corrected mode is active in the next sawing cycle. (The synchronising factor, as it has been entered here, is entered the same way into axis register R446 "Scaling Factor Slave" of the respective table for the sawing axis) For this, please refer to chapter 7.3 "Adjustment of Synchronous Operation“ on page 26 and chapter 9.8.12 "MotionFS_SetSyncCorrection“ on page 62.

Default value: 1 = Synchronising Factor 1:1
Value range: FLOAT limits,
< 1 = The sawing axis is running slower than defined in the table
> 1 = The sawing axis is running faster than defined in the table

Reserved0: Reserved for further extensions

**Description**

Definition of data for a Flying Saw axis. The data are input, managed and used by the Flying Saw functions. For each available Flying Saw axis, a variable of this structure has to be declared.

The controller program unit on the user side only accesses the parameters CuttingPointCorr and SyncFactor, if a cutting position correction is to be carried out, or if axes are to be synchronized.

Some parameters contain a value which is only valid in the respective sawing cycle, e.g. ActMasterSyncPos. These parameters are updated, when one of the functions for cycle start, such as MotionFS_Start, MotionFS_StartCapPos, MotionFS_StartChangeTab or MotionFS_StartInstant is called up. The parameter LastTabOffsM is updated synchronously.

### 9.5.2 MOTION_FS_TAB_DATA

**Definition**

```
MOTION_FS_TAB_DATA:
    STRUCT
        TabNum, SyncPosTabIdx, MasterSyncPos, MasterPosTabMin, SlaveSyncPos,
        SlavePosTabMin, TabStartIdx, TabGearRatio, Reserved0: FLOAT;
    END_STRUCT;
```

**Parameters**

TabNum: Table configuration number

SyncPosTabIdx: Index of the table element with the synchronous positions. The synchronous position is the position, as of which the Flying Saw axis is synchronous with the master axis.

MasterSyncPos: The synchronous position of the master in mm, respectively °

MasterPosTabMin: The first position of the master in mm, respectively °

SlaveSyncPos: The synchronous position of the slave in mm, respectively °

SlavePosTabMin: The first table position of the slave in mm, respectively °

| | |
|---|---|
| TabStartIdx: | Index to first table element |
| TabGearRatio: | Table transmission ratio between motion master and motion slave Here, the value of the table transmission ratio TabGearRatio is stored, which is handed over to the function MotionFS_LoadTab_Linear, respectively MotionFS_LoadTab_Sin2 when calling them up. |
| Reserved0: | Reserved for further extensions |

**Description**

Definition of data to be compiled in a table in which the acceleration and synchronous motion values of a Flying Saw are set. The data are input, managed and used by the Flying Saw functions. For each table definition of a Flying Saw axis, a variable of this structure has to be declared.

### 9.5.3 MOTION_FS_TMP_FLOATVARS

**Definition**

```
MOTION_FS_TMP_FLOATVARS:
        STRUCT
                Tmp0, Tmp1, Tmp2, Tmp3, Tmp4, Tmp5, Tmp6, Tmp7: FLOAT;
        END_STRUCT;
```

**Parameters**

| | |
|---|---|
| Tmp0...Tmp7: | Temporary FLOAT variables |

**Description**

Definition of the temporary FLOAT variables which are used as FLOAT buffers for calculations of Flying Saw functions. When the Flying Saw functions for which a variable of this structure is needed, are called up simultaneously in various tasks, as many variables of this structure are needed, as there are simultaneous function calls.

## 9.6 Constants

| Survey of Constants |
|---|
| **For the function MotionFS_StartInstant:** |
| **c_FS_STARTINSTANT_TIME_OF_PROCESS** = 100<br>Dead time of the starting process for an immediate cut in ms. |
| **For the function MotionFS_Ini:** |
| **c_FS_INI_STAT_OK** = 0<br>Execution ok |
| **c_FS_INI_STAT_ERR_MASTER_ENABLED** = -1<br>Error: The travel range of the motion master cannot be set, as the output stage of the axis has been activated. The output stage must be deactivated. The error can only occur, when AutoCalcMasterPosLimit = 1 has been set. |
| **c_FS_INI_STAT_ERR_SLAVE_ENABLED** = -2<br>Error: The master position range of the Flying Saw axis cannot be set, as the output stage of the axis has been activated. The output stage must be deactivated. The error can only occur, when AutoCalcMasterPosLimit = 1 has been set. |
| **c_FS_INI_STAT_ERR_MASTER_POSRANGE_TOO_SMALL** = -3<br>Error: The travel range of the motion master is too small. The travel range must be twice as big as the set greatest possible cutting length MaxCuttingLength. |
| **For the function MotionFS_LoadTab_Linear / MotionFS_LoadTab_Sin2:** |
| **c_FS_LOADTAB_STAT_OK** = 0<br>Execution ok |
| **c_FS_LOADTAB_STAT_ERR_ACC_PHASE_TOO_LONG** = -1<br>Error: The acceleration distance is longer than the cutting length that is to be checked.<br>pTmp7 = 0: Acceleration distance ><br>MOTION_FS_SLAVE_DATA.MaxCuttingLength<br>pTmp7 > 0: Acceleration distance > pTmp7 |
| **c_FS_LOADTAB_STAT_WARN_REMAINTIME_TOO_SMALL** = 10<br>Danger: The remaining cycle time from the end of the synchronizing phase to the end of the table is too small.<br>pTmp6 = 0: remaining cycle time < 1 ms<br>pTmp6 > 0: remaining cycle time < pTmp6 |
| **For the function MotionFS_Start / MotionFS_StartChangeTab:** |
| **c_FS_START_STAT_OK** = 0<br>Execution ok |
| **c_FS_START_STAT_ERR_MASTER_POS_TOO_FAR** = -1<br>Error: Starting the cycle is not possible, as the master position is already greater than the starting position determined internally. |
| **For the function MotionFS_StartInstant / MotionFS_StartCapPos:** |
| **c_FS_STARTINSTANT_STAT_OK** = 0<br>Execution ok |
| **For the function MotionFS_StartStatus:** |

| |
|---|
| **c_FS_STARTSTATUS_STAT_NO_TABLE_ACTIVE** = 0<br>The table function is not active. |
| **c_FS_STARTSTATUS_STAT_WAITING_FOR_START** = 1<br>The Flying Saw delays, until the master position has reached the internally determined starting position. |
| **c_FS_STARTSTATUS_STAT_IN_RAMP** = 2<br>The sawing axis is in the synchronizing phase. |
| **c_FS_STARTSTATUS_STAT_SYNCHRON** = 3<br>The sawing axis is synchronous with the master axis. |
| **c_FS_STARTSTATUS_STAT_ERR_INCORRECT_STATUS** = -1<br>A regular status is not recognized: Although the table has been started, a reasonable status for a Flying Saw application could be recognized. |

### 9.6.1 c_FS_STARTINSTANT_TIME_OF_PROCESS

This constant is exclusively used by the function MotionFS_StartInstant. At executing the function MotionFS_StartInstant there arises a dead time from calling up the function up to the moment, at which the function issues command 46 for starting the table mode for the sawing axis.

During this dead time, the motion master covers a certain distance. The Function MotionFS_StartInstant must calculate this distance. The dead time depends on the cycle time of the controller and on the workload of the system bus. The same way, minor master speeds (< 10 revolutions/min) have got a greater influence on the calculated distance covered by the motion master than greater master speeds.

In this case, the constant c_FS_STARTINSTANT_TIME_OF_PROCESS specifies an approximate dead time value in ms.

Important

**If a value smaller than the actual dead time is selected, the head cut is not carried out immediately. At table function activated, the sawing axis delays, until the master has reached the starting position for table mode again. This does not happen, unless the master has carried out a complete modulo run.**

If the value is too great, there might occur an undesired delay time, until the head cut is carried out.

The deafault value of 100 ms is a practical value leaving a great enough difference to the dead time and is therefore apt for minor speeds. If there results an undesired delay time, especially for greater speeds, the value can be reduced. Yet, the value should not be set to less than 10 ms.

## 9.7 Standard Variable Declaration for All Sample Programs

The following variable declaration applies to all sample programs, respectively program parts of this document.

```
VAR
     ...
     st_MotionFS_Slave0Data:         MOTION_FS_SLAVE_DATA AT %VL 65024;
     st_MotionFS_Slave0TabData0:     MOTION_FS_TAB_DATA AT  %VL 65036;
     st_MotionFS_Slave0TabData1:     MOTION_FS_TAB_DATA AT  %VL 65045;
     st_MotionFS_TmpFloatVars:       MOTION_FS_TMP_FLOATVARS AT %VL 65054;
     ...
     f_MasterSpeed:                  FLOAT AT %VL 65073;
     f_SlaveAcc:                     FLOAT AT %VL 65074;
```

```
     f_SlaveStartPos:               FLOAT AT %VL 65076;
     f_TabGearRatio:                FLOAT AT %VL 65077;
     f_CuttingLength:               FLOAT AT %VL 65070;
     ...
     n_Return:                      INT AT %VL 200;
     ...
VAR_END;
```

## 9.8 Functions

| Overview of Functions |
|---|
| **Functions being carried out during the initialization phase of the controller program:** |
| **MotionFS_Ini** (pMasterBase, pSlaveBase, pSlaveData, AutoCalcMasterPosLimit, pMaxCuttingLength, Option, pTmp0)<br>Main function for initializing the Flying Saw application |
| **MotionFS_LoadTab_Linear** (pSlaveData, pTabData, pSlaveStartPos, pMinCuttingLength, pMaxMasterSpeed, pMaxAcc, TabPoints, TabNum, TabStartIdx, pTmp0)<br>Calculates and loads a Flying Saw table with linear speed ramp into the sawing axis. |
| **MotionFS_LoadTab_Sin2** (pSlaveData, pTabData, pSlaveStartPos, pMinCuttingLength, pMaxMasterSpeed, pMaxAcc, TabPoints, TabNum, TabStartIdx, pTmp0)<br>Calculates and loads a Flying Saw table with sine-square speed ramp to the sawing axis. |
| **Functions for Starting and Carrying Out a Sawing Cycle:** |
| **MotionFS_Reset** (pSlaveData, pTabData, Option)<br>Initializes the start of the next production period |
| **MotionFS_Start** (pSlaveData, pTabData, pCuttingLength, Option, pTmp0)<br>Starts a new sawing cycle for a classical Flying Saw. |
| **MotionFS_StartChangeTab** (pSlaveData, pTabDataNext, pTabDataLast, pCuttingLength, Option, pTmp0)<br>Starts a new sawing cycle for a classical Flying Saw with changeover from one table to another. |
| **MotionFS_StartInstant** (pSlaveData, pTabData, Option, pTmp0)<br>Starts a head cut. |
| **MotionFS_StartCapPos** (pSlaveData, pTabData, Option, pTmp0)<br>Starts a new sawing cycle for a Flying Saw by hardware signal. |
| **MotionFS_SetSyncCorrection** (pSlaveData, pCorrection, Option)<br>Sets a new synchronous correction value for the next sawing cycle. |
| **Functions for Determining Important Parameters:** |
| **MotionFS_GetStartStatus** (pSlaveData, Option)<br>Status feedback at starting a new sawing cycle |
| **MotionFS_GetActCuttingLength** (pSlaveData, Direction, Option, pTmp0, pActCuttingLength)<br>Feedback of the length in mm, respectively °, which the material is still to cover before reaching the synchronized position. For sawing cycles started by the function MotionFS_Start or MotionFS_StartChangeTab. |

| |
|---|
| **MotionFS_GetDistToCutPoint** (pSlaveData, Option, pTmp0, pDistCutPoint)<br>Feedback of the length in mm, respectively °, which the material is still to cover before reaching the synchronized position. For sawing cycles started by the function MotionFS_StartInstant or MotionFS_StartCapPos. |
| **MotionFS_GetMasterAlignPos** (pTabData, Option, pTmp0, pMasterAlignPos)<br>Reports back the position in mm, respectively °, to which the master has to be set, isf the next production period without head cut, but with synchronized material and tools unit is to be started. |

### 9.8.1 MotionFS_Ini

**MotionFS_Ini** (pMasterBase, pSlaveBase, pSlaveData, AutoCalcMasterPosLimit, pMaxCuttingLength, Option, pTmp0)

**Parameters**

pMasterBase: Start address of the JM-2xx at the system bus which is to be the motion master, e.g., 12000; this represents the first JM-2xx directly following the controller. If the master is a JX2-CNT1, the pMasterBase must equal zero.

pSlaveBase: Start address of the JM-2xx at the system bus which is to be the sawing axis, e.g., 13000; this represents the second JM-2xx directly following the controller.

pSlaveData: Start address of the MOTION_FS_SLAVE_DATA structure which is to be loaded with data of the specified sawing axis.

AutoCalcMasterPosLimit: Switch to define whether the master position range in the motion master and in the sawing axis is to be calculated and set automatically, see function description.

Value range: 0;1

0 = Master position range is set by the user
1 = The master position range is set automatically

pMaxCuttingLength: Address of the FLOAT-variable containing the greatest possible cutting length in mm respectively °, see function description.

Value range of the cutting length: 0...250000 mm

Option: Reserved for expansions: Always = 0

pTmp0: Start address of the structure of the temporary FLOAT-variable, type MOTION_FS_TMP_FLOATVARS.

**Return Value** Function Status The possible values are
c_FS_INI_STAT_OK
c_FS_INI_STAT_ERR_MASTER_ENABLED
c_FS_INI_STAT_ERR_SLAVE_ENABLED
c_FS_INI_STAT_ERR_MASTER_POSRANGE_TOO_SMALL
Please also refer to chapter 9.6 “Constants” on page 36

**Description**

The function describes the MOTION_FS_SLAVE_DATA structure of the respective sawing axis with initial values and automatically sets the master position range which depends on the switch AutoCalcMasterPosLimit.

AutoCalcMasterPosLimit

Please refer to chapter 2.2 „Setting the Master Positioning Range in the Slave“ on page 5.

0 = travel range (axis register R182, R183) in the motion master and the master position range (axis registers R158, R159) in the sawing axis are not changed. Travel range and master position range are considered to have been set correctly by the user already. The master position range must be twice as big as the set greatest possible cutting length MaxCuttingLength.
If this is not the case, the function is terminated by the error message c_FS_INI_STAT_ERR_MASTER_POSRANGE_TOO_SMALL.

1 = Travel range (axis registers R182, R183) of the motion master (in case the motion master is not a JX2-CNT1), and the master position range (axis registers R158, R159) of the sawing axis are automatically set dependent on the greatest possible cutting length MaxCuttingLength. Which position range is automatically set then, can be found in Fig. 1 in chapter 2.2 „Setting the Master Positioning Range in the Slave“ on page 5.

For automatically setting the travel range and the master position range, the output stage of the motion master and of the sawing axis must be blocked. If this is not the case, the function is terminated by the error message c_FS_INI_STAT_ERR_MASTER_ENABLED or
it is terminated by c_FS_INI_STAT_ERR_SLAVE_ENABLED.

pMaxCuttingLength

In case of classical Flying Saws, the greatest possible cutting length is entered here, which might occur during the respective application. In case of Flying Saws with hardware signal, the maximum distance between the hardware signals must be specified here.

During operation, other cutting lengths / distances are possible. They can be significantly smaller, yet, they must never be greater than the MaxCuttingLength specified here. The cutting length / distance is related to the position scaling of the motion master.

Processing the Function

The function is being processed during the initialization phase of the controller program, before any othe Flying Saw function is carried out. It must be processed in the respective MOTION_FS_SLAVE_DATA structure for each sawing axis belonging to the system individually.

**Important**

**Before configuring the axis groups for master-slave operation, the function must be called up; please also refer to chapter 2.1 Steps and Sequence of the Initialisation on page 5.**

Handling the Axis Registers

The following axis registers are described in the motion master, if AutoCalcMasterPosLimit = 1: R182, R183. The following axis registers are described for the sawing axis: R435, R436, and if AutoCalcMasterPosLimit = 1: R158, R159.

**Example**

In the following example, the motion master has been defined in JetSym as first axis, while the motion slave (= sawing axis) has been defined as second axis. The axes have been connected to a JC-24x. In the motion master and the sawing axis, the positioning range is to be set automatically. The maximum possible cutting length to occur after initializing the controller program for this sawing axis amounts to 5000 mm.

```
TASK t_Init
     ...
     //--- Carry out FS commissioning for slave 0
     f_CuttingLength:= 5000;     //--- Maximum Cutting Length
     n_Return:=MotionFS_Ini(12000,13000,&st_MotionFS_Slave0Data,1,
                        &f_CuttingLength,0,&st_MotionFS_TmpFloatVars);
     //--- Evaluating the return value
     CASE n_Return OF
             c_FS_INI_STAT_ERR_MASTER_ENABLED:
             //--- Error: Motion master has been released
             ...
             BREAK;
         c_FS_INI_STAT_ERR_SLAVE_ENABLED:
             //--- Error: Sawing axis has been released
             ...
             BREAK;
         c_FS_INI_STAT_ERR_MASTER_POSRANGE_TOO_SMALL:
             //--- Error: The master position range is too small
             ...
             BREAK;
     END_CASE;

     //--- Configuration of the axis group
     ...
END_TASK;
```

### 9.8.2 MotionFS_LoadTab_Linear / MotionFS_LoadTab_Sin2

**MotionFS_LoadTab_Linear** (pSlaveData, pTabData, pSlaveStartPos, pMaxMasterSpeed, pMaxAcc, pTabGearRatio, TabPoints, TabNum, TabStartIdx, pTmp0)

**MotionFS_LoadTab_Sin2** (pSlaveData, pTabData, pSlaveStartPos, pMaxMasterSpeed, pMaxAcc, pTabGearRatio, TabPoints, TabNum, TabStartIdx, pTmp0)

**Parameters**

pSlaveData: Start address of the structure, type MOTION_FS_SLAVE_DATA for the sawing axis, to which the table is to be loaded.

pTabData: Start address of the MOTION_FS_TAB_DATA structure which is to be loaded with data of the table to be configured.

pSlaveStartPos: Address of the FLOAT-variable containing the starting position of the sawing axis in mm, respectively °, see function description.

Value range of the starting position: Travel range of the sawing axis

pMaxMasterSpeed: Address of the FLOAT variable, into which the maximum master speed has been entered in mm/s, respectively °/s, by which the table to be configured is processed, see function description.

Value range of the speed: 0...positive FLOAT limit

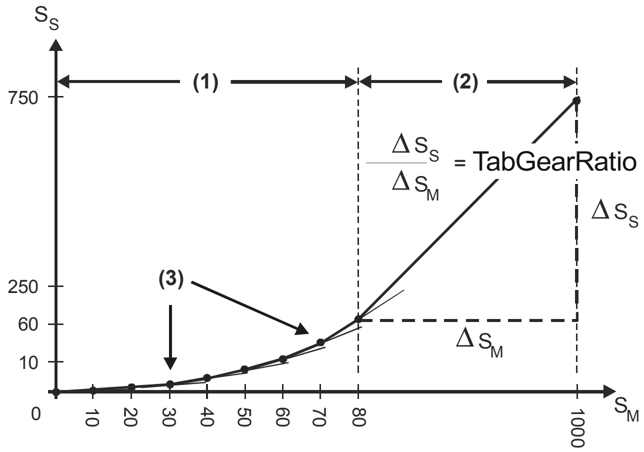
| | |
|---|---|
| pMaxAcc: | Address of the FLOAT variable, into which the maximum acceleration of the master has been entered in mm/s², respectively °/s², by which the sawing axis is to be processed during the synchronous phase, see function description.<br><br>Value range of the acceleration: 0...positive FLOAT limit |
| pTabGearRatio: | Address of the FLOAT-variable, in which the table transmission ratio for the synchronous phase between the master position and the slave position is set, see function description.<br><br>Value range of the transmission ratio: FLOAT range<br>Normally: 1 (=master-slave transmission 1:1) |
| TabPoints: | Number of points the table to be configured is to have, see function description.<br><br>Value range: 10...4096<br>Recommended value range: 20 – 50<br>Minimum value: 10 |
| TabNum: | Table configuration number for the table to be configured<br><br>Value range: 0...23 |
| TabStartIdx: | Starting index referring to the table element, from which the points of the table to be configured are to be entered, see function description.<br><br>Value range 0...4095 |
| pTmp0: | Start address of the structure of the temporary FLOAT-variable, type MOTION_FS_TMP_FLOATVARS, see function description. |

**Return Value**

Function Status The possible values are
c_FS_LOADTAB_STAT_OK
c_FS_LOADTAB_STAT_ERR_ACC_PHASE_TOO_LONG
c_FS_LOADTAB_STAT_WARN_REMAINTIME_TOO_SMALL
Also refer to the parameter pTmp0
Please also refer to chapter 9.6 “Constants” on page 36

**Description**

The function is to configure a table for the sawing axis, which is to define the acceleration and synchronous motion between motion master and the sawing axis. Dependent on which function is being used, a linear or sine-square speed ramp for the acceleration motion of the sawing axis will result. In this case, the respective MOTION_FS_TAB_DATA structure is described.




($S_S$) Position of the sawing axis
($S_M$) Position of the motion master
(1) Synchronizing phase
(2) Synchronous phase
(3) Table points (in the illustration: TabPoints = 10)

**Fig. 22 Illustration of the table points in the master-slave diagram**

The table position range of the motion master starts in the middle of the master position range that has been set for the sawing axis. It has got exactly the length of the greatest possible cutting length MaxCuttingLength that has been specified in the function MotionFS_Ini. Please also refer to chapter 2.2 „Setting the Master Positioning Range in the Slave" on page 5. The table position range for the sawing axis starts at the starting position SlaveStartPos. The end of the table position range for the sawing axis is dependent on the end of the table position range of the motion master and of the specified table transmission ratio TabGearRatio, see Fig. 22.

**Important**

**The sawing axis must by no means reach the end of the table position rrange, as it would come to an abrupt halt there. Before reaching the end of the table position range, it must be decoupled from the table function; for this, please refer to chapter 3.6 „Disengaging" on page 10. The table length (length of the master positioning range of the table) is identical with the value of the parameter MaxCuttingLength, which is transferred at calling up the function MotionFS_Ini.**

pSlaveStartPos

In general, the sawing axis starts at the starting position. During operation, the sawing axis can also be started from any other position, without re-configuring the table by a new starting position. Yet, there are various aspects to be considered; for this, please refer to chapter 6.1 „Change the Starting Position of the Sawing Axis" on page 22.

pMaxMasterSpeed / pMaxAcc

During operation, the actual master speed need not be exactly the same as the set value. The acceleration set for the sawing axis will only be reached, though, if the motion master is running by the set speed exactly. The lower the actual master speed is compared to the set speed, the lower is the acceleration of the sawing axis and vice versa.

If the master axis is to run at a speed that significantly differs from the set speed, a new table has to be configured, in order for the sawing axis largely to reach the set acceleration ratio.

pTabGearRatio

Normally, the table transmission ratio is 1. For certain applications, though, the sawing axis might have to move faster or slower than the motion master during the synchronous phase; please refer to chapter 6.3 „Speed Difference between Sawing Axis and Motion Master“ on page 23.

TabPoints

The greater the amount of table points, the finer is the resolution of the synchronizing phase. The synchronizing phase takes up all table points except the last one; see Fig. 22.
The possible maximum value depends on the number of tables to be configured. Altogether, there are 4096 table elements available for all configured tables; for this, please refer to the table description in the document "jm2xx_at_jetcontrol_bi_xxxx_user_information.pdf".

TabStartIdx

If several tables are to be configured, please make sure the table sections do not overlap. Next free table element for another table: TabStartIdx of the table applied last + TabPoints of the table applied last.

pTmp0

The function compares the distance of the acceleration travel with the minimum cutting length, which is transmitted via variable Tmp7 in mm, respectively °. If the acceleration distance is longer than this minimum cutting length, the function is interrupted by the error message c_FS_LOADTAB_STAT_ERR_ACC_PHASE_TOO_LONG. In this case, operation by the minimum cutting length at the set acceleration, respectively master speed, is not possible. Either the acceleration ratio must be increased, or the master speed must be decreased. If Tmp7 = 0, the greatest possible cutting length MaxCuttingLength specified in the function MotionFS_Ini is used for a comparison.

Further, the function compares the remaining time with a set value, which is transmitted within a few seconds by means of variable Tmp6. The remaining time is the time which is still available for the sawing axis after reaching the synchronized position for carrying out the cut and returning to the starting position then. If the remaining time is smaller than the set value in Tmp6, the function will report the warning c_FS_LOADTAB_STAT_WARN_REMAINTIME_TOO_SMALL. The function is still processed correctly and thus configures a correct table.
If Tmp6 = 0, a comparison with a standardized set value of 1 s is made. Basically, at correct function processing, the actual remaining time is fed back via variable Tmp6.

Processing the Function

The function can be processed any time after processing the function MotionFS_Ini for the sawing axis. The dynamic table definition, that is, making a definition during operation, can be time-critical. For this reason, it is recommended to initialize the tables during the initialization phase of the controller program.

Handling the Axis Registers

The following axis registers are described for the sawing axis: Table configuration: R410, R411, R413, R410 R411, R413, R443, R446, R440, R441, R442.

**Example**

A table of a linear speed ramp and 50 table points is to be set up. The speed of the master axis is approximately 500 mm/s, while the sawing axis is to accelerate by 1000 mm/s² to the synchronized position. In this example, the sawing axis is to start from position 2500. Further, the sawing axis is as fast as the motion master during the synchronized phase. This means the transmission ratio in the table between motion master and sawing axis is 1:1.

As there already exists a table of configuration number 0 and 50 table points, table configuration 1 is to be applied to the table.

The greatest possible cutting length is 5000 mm and has alredy been set for the function MotionFS_Ini. During operation, the cutting length can change. The smallest possible cutting length is 3500 mm. For the function to check whether the acceleration distance exceeds the minimum cutting length, the cutting length is specified in millimeters in st_MotionFS_TmpFloatVars.Tmp7.

For the cut of the maximum cutting length, about 5 seconds for cut and homeward voyage (except acceleration) are needed. For the function to check whether this time is acceptable, this time is specified in seconds in st_MotionFS_TmpFloatVars.Tmp6.

```
TASK t_Init
      ...
      //--- Setting the framework values
      f_SlaveStartPos:= 2500;
      f_MasterSpeed:= 500;
      f_SlaveAcc:= 1000;
      f_TabGearRatio:=1;  //--- Transmission ratio between motion master and sawing axis
                          //= 1:1
      st_MotionFS_TmpFloatVars.Tmp6:= 5;      //--- Set value of remaining time
      st_MotionFS_TmpFloatVars.Tmp7:= 3500;   //--- Minimum cutting length

      //--- Carrying out the function
      n_Return:=MotionFS_LoadTab_Linear(&st_MotionFS_Slave0Data,
      &st_MotionFS_Slave0TabData1,&f_SlaveStartPos,&f_MasterSpeed,&f_SlaveAcc,
      &f_TabGearRatio,50,1,50,&st_MotionFS_TmpFloatVars0);

      //--- Evaluating the return value
      CASE n_Return OF
          c_FS_LOADTAB_STAT_WARN_REMAINTIME_TOO_SMALL:
              //--- Warning: Remaining time is smaller than the set value
              ...
              BREAK;
          c_FS_LOADTAB_STAT_ERR_ACC_PHASE_TOO_LONG:
              //--- Error: Acceleration distance is longer than the minimum cutting length
              ...
              BREAK;
      END_CASE;
      ...
 END_TASK;
```

### 9.8.3 MotionFS_Reset

**MotionFS_Reset** (pSlaveData, pTabData, Option)

**Parameters**

pSlaveData: Start address of the structure, type MOTION_FS_SLAVE_DATA for the sawing axis, for which the new production period is to be started.

pTabData: Start address of the structure of the type MOTION_FS_TAB_DATA containing the data of the table by which the new production period is to be started.

Option: Reserved for expansions: Always = 0

**Return Value** No return value

**Description**

Initializes the Flying Saw functions for a new production period. For a definition of the term "production period", please refer to chapter 3 „Production Period and Sawing Cycle“ on page 8.

<u>Processing the Function</u>

**Important**

**This function MUST be called up before any new production period. For this, the structure MOTION_FS_TAB_DATA must be applied, which will also be used for the following starting function (MotionFS_Start, MotionFS_StartCapPos, etc.).**

<u>Handling the Axis Registers</u>

The following axis registers are described for the sawing axis: R101, R410, R443, R412, R448, R449.

**Example**

```
TASK t_Run
      ...
      //--- Start a new production period
          MotionFS_Reset(&st_MotionFS_Slave0Data,&st_MotionFS_Slave0TabData0,0);
          ...
      //--- Carry out the sawing cycles
      WHILE ... DO
          ...
          //--- Start the sawing cycle
              MotionFS_Start...(...);
              ...
      END_WHILE;
      ...
END_TASK;
```

### 9.8.4 MotionFS_Start

**MotionFS_Start** (pSlaveData, pTabData, pCuttingLength, Option, pTmp0)

**Parameters**

pSlaveData: Start address of the structure, type MOTION_FS_SLAVE_DATA, for the sawing axis, for which a new sawing cycle is to be started.

pTabData: Start address of the structure of the type MOTION_FS_TAB_DATA containing the data of the table by which the new sawing cycle is to be started.

pCuttingLength: Address of the FLOAT-variable containing the set cutting length in

| | |
|---|---|
| | mm respectively °, see function description.<br><br>Value range of the cutting length: 0...table position range of the motion master |
| Option: | Reserved for expansions: Always = 0 |
| pTmp0: | Start address of the structure of the temporary FLOAT-variable, type MOTION_FS_TMP_FLOATVARS. |

**Return Value** Function Status The possible values are
c_FS_START_STAT_OK
c_FS_START_STAT_ERR_MASTER_POS_TOO_FAR
Please also refer to chapter 9.6 “Constants” on page 36

**Description**

The function starts a new sawing cycle for a classical Flying Saw axis.

Before the first sawing cycle that is started by this function, either a head cut by means of the function MotionFS_StartInstant must have preceded, or else the material must have been synchronized with the sawing unit. If this is not the case, the cutting position is not „referenced“. For synchronizing, the function MotionFS_GetMasterAlignPos can be applied.

The return value must be evaluated at each call-up, in order to detect possible errors.

The function reduces the actual position of the motion master by the cutting length and calculates a new starting position for the sawing axis. If the reduced master position has already exceeded the starting position, the function is terminated, while the following error message is issued: c_FS_START_STAT_ERR_MASTER_POS_TOO_FAR. If the master axis has not reached the starting position yet, the function starts processing the table for the sawing axis in the following mode: "Delay, until the master position has reached the starting position". The sawing axis is then started by the table function in the axis amplifier. After this, the function checks, if the sawing axis has reached the synchronizing position.

**If no error has occurred, the function will not be terminated until the sawing axis and the motion master have been synchronized.**

**Important**

pCuttingLength

As set cutting length, any value can be specified; yet, this value must not be greater than the greatest possible cutting length pMaxCuttingLength that has been specified at calling up the function MotionFS_Ini. The set cutting length is related to the position scaling of the motion master.

Processing the Function

The function has to be processed for each new sawing cycle.

Handling the Axis Registers

The following axis registers are described for the sawing axis: R101, R402, R410, R443, R446.

**Example**

```
TASK t_Run
      ...
      //--- Carry out the sawing cycles
```

```
    WHILE ... DO
        ...
        //--- Move the saw to the starting position
            ...
        //--- Start the sawing cycle
        f_CuttingLength:= 5000;
        MotionFS_Start(&st_MotionFS_Slave0Data,&st_MotionFS_Slave0TabData0,
                    &f_CuttingLength,0,&st_MotionFS_TmpFloatVars0);
        //--- Carry out the cut
            ...
    END_WHILE;
    ...
END_TASK;
```

### 9.8.5 MotionFS_StartChangeTab

**MotionFS_StartChangeTab** (pSlaveData, pTabDataNext, pTabDataLast, pCuttingLength, Option, pTmp0)

**Parameters**

| | |
|---|---|
| pSlaveData: | Start address of the structure, type MOTION_FS_SLAVE_DATA, for the sawing axis, for which a new sawing cycle is to be started. |
| pTabDataNext: | Start address of the structure of the type MOTION_FS_TAB_DATA containing the data of the table by which the new sawing cycle is to be started. |
| pTabDataLast: | Start address of the structure of the type MOTION_FS_TAB_DATA containing the data of the table by which the latest sawing cycle was started. |
| pCuttingLength: | Address of the FLOAT-variable containing the set cutting length in mm respectively °, see function description.<br><br>Value range of the cutting length: 0...table position range of the motion master |
| Option: | Reserved for expansions: Always = 0 |
| pTmp0: | Start address of the structure of the temporary FLOAT-variable, type MOTION_FS_TMP_FLOATVARS. |

**Return Value**

Function Status The possible values are
c_FS_START_STAT_OK
c_FS_START_STAT_ERR_MASTER_POS_TOO_FAR
Please also refer to chapter 9.6 “Constants” on page 36

**Description**

The function starts a new sawing cycle for a classical Flying Saw axis. In this case, the cut to follow is to be carried out by another table than the latest one.

A cut has to be carried out by means of the function MotionFS_StartInstant or MotionFS_Start and another table before the sawing cycle is started by above mentioned function.

The return value must be evaluated at each call-up, in order to detect possible errors.

The function reduces the actual position of the motion master by the cutting length and calculates a new starting position for the sawing axis. If the reduced master position has already exceeded the starting position, the function is terminated, while the following error message is issued:

c_FS_START_STAT_ERR_MASTER_POS_TOO_FAR. If the master axis has not reached the starting position yet, the function starts processing the table for the sawing axis in the following mode: "Delay, until the master position has reached the starting position". The sawing axis is then started by the table function in the axis amplifier. After this, the function checks, if the sawing axis has reached the synchronizing position.

**If no error has occurred, the function will not be terminated until the sawing axis and the motion master have been synchronized.**

**Important**

<u>pCuttingLength</u>

As set cutting length, any value can be specified; yet, this value must not be greater than the greatest possible cutting length pMaxCuttingLength that has been specified at calling up the function MotionFS_Ini. The set cutting length is related to the position scaling of the motion master.

<u>Processing the Function</u>

The function is called up for each new sawing cycle requiring another table than the one that was used at the latest cut.

<u>Handling the Axis Registers</u>

The following axis registers are described for the sawing axis: R101, R402, R410, R412, R443, R446.

**Example**

In the following example, a start applying table 0 or table 1 is initiated, dependent on a switch giving the CASE instruction. For changeover between tables, the function MotionFS_StartChangeTab has to be applied.

```
TASK t_Run
     ...
     //--- Carry out the sawing cycles
     WHILE ... DO
             ...
         //--- Move the saw to the starting position
             ...
         //--- Start the sawing cycle
             f_CuttingLength:= 5000;
         //--- Selecting the table
         CASE ...OF

             c_Tab0: //--- Start by Tab0, if Tab0 has already been used
                     MotionFS_Start(&st_MotionFS_Slave0Data,&st_MotionFS_Slave0TabData0,
                                     &f_CuttingLength,0,&st_MotionFS_TmpFloatVars0);

             c_Tab0ToTab1: //--- Change to Tab1, if Tab0 has already been used
                     MotionFS_StartChangeTab(&st_MotionFS_Slave0Data,
                             &st_MotionFS_Slave0TabData1,&st_MotionFS_Slave0TabData0,
                                     &f_CuttingLength,0,&st_MotionFS_TmpFloatVars0);

             c_Tab1: //--- Start by Tab1, if Tab1 has already been used
                     MotionFS_Start(&st_MotionFS_Slave0Data,&st_MotionFS_Slave0TabData1,
                                     &f_CuttingLength,0,&st_MotionFS_TmpFloatVars0);

             c_Tab1ToTab0: //--- Change to Tab0, if Tab1 has already been used
                     MotionFS_StartChangeTab(&st_MotionFS_Slave0Data,
                             &st_MotionFS_Slave0TabData0,&st_MotionFS_Slave0TabData1,
                                     &f_CuttingLength,0,&st_MotionFS_TmpFloatVars0);
         END_CASE;
```

```
        //--- Carry out the cut
            ...
    END_WHILE;
    ...
END_TASK;
```

### 9.8.6 MotionFS_StartInstant

**MotionFS_StartInstant** (pSlaveData, pTabData, Option, pTmp0)

**Parameters**

| | |
|---|---|
| pSlaveData: | Start address of the structure, type MOTION_FS_SLAVE_DATA for the sawing axis, for which an immediate sawing cycle is to be started. |
| pTabData: | Start address of the structure of the type MOTION_FS_TAB_DATA containing the data of the table by which the new sawing cycle is to be started. |
| Option: | Reserved for expansions: Always = 0 |
| pTmp0: | Start address of the structure of the temporary FLOAT-variable, type MOTION_FS_TMP_FLOATVARS. |

**Return Value** Function Status The possible values are
c_FS_START_STAT_OK

**Description**

The function starts a new sawing cycle by a head cut When the function is called up, only the run time of the function is needed, until the cut is carried out. Please refer to chapter 9.6.1 „c_FS_STARTINSTANT_TIME_OF_PROCESS“ on page 37.

**Important** **If the value for c_FS_STARTINSTANT_TIME_OF_PROCESS smaller than the actual dead time is selected, the head cut is not carried out immediately. At table function activated, the sawing axis delays, until the master has reached the starting position for table mode again. This does not happen, unless the master has carried out a complete modulo run.**

As the start is carried out via the controller program, the process includes a delay time that cannot be exactly exactly between function start and cutting instance. Therefore, this function is not apt for optimised trimming. The function is used for carrying out the head cut of a new production period in classical Flying Saw applications. As the head cut produces wastage any way, cutting inaccuracies can be accepted in this application.

After the head cut, the material is "referenced". For this reason, the function MotionFS_Start or MotionFS_StartChangeTab is used for the sawing cycles to follow.

**If no error has occurred, the function will not be terminated until the sawing axis and the motion master have been synchronized.**

**Important**

<u>Processing the Function</u>

The function is called up at each start of a new production period, when a head cut for a classical Flying Saw function is to be made.

Handling the Axis Registers

The following axis registers are described for the sawing axis: R101, R402, R410, R443, R446.

**Example**

```
TASK t_Run
     ...
     //--- Start a new production period
         MotionFS_Reset(&st_MotionFS_Slave0Data,&st_MotionFS_Slave0TabData0,0);

     //--- Carry out a head cut first
         MotionFS_StartInstant(&st_MotionFS_Slave0Data,&st_MotionFS_Slave0TabData0,
                               0,&st_MotionFS_TmpFloatVars0);

     //--- Carry out the sawing cycles
     WHILE ... DO
         //--- Move the saw to the starting position
             ...
         //--- Start the sawing cycle
             MotionFS_Start...(...);
         //--- Carry out the cut
             ...
     END_WHILE;
     ...
 END_TASK;
```

### 9.8.7 MotionFS_StartCapPos

**MotionFS_StartCapPos** (pSlaveData, pTabData, pCapPos, pPosOffset, Option, pTmp0)

**Parameters**

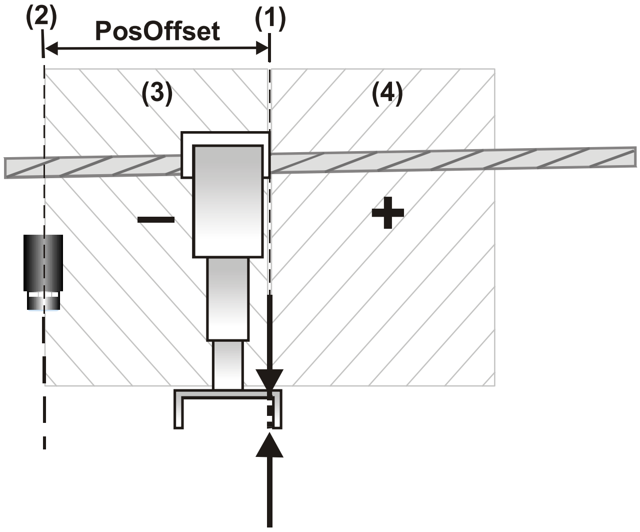
| | |
|---|---|
| pSlaveData: | Start address of the structure, type MOTION_FS_SLAVE_DATA for the sawing axis, for which the sawing cycle is to be started. |
| pTabData: | Start address of the structure of the type MOTION_FS_TAB_DATA containing the data of the table by which the new sawing cycle is to be started. |
| pCapPos: | Address of the FLOAT-variable containing the positon value in mm respectively ° of the motion master at the instance of capturing; see function description.<br><br>Value range of the capture position: Master position range |
| pPosOffset: | Address of the FLOAT-variable containing the distance of the position offset in mm respectively °, see function description.<br><br>Value range of the offset: Master position range |
| Option: | Reserved for expansions: Always = 0 |
| pTmp0: | Start address of the structure of the temporary FLOAT-variable, type MOTION_FS_TMP_FLOATVARS. |

**Return Value** Function Status The possible values are
c_FS_START_STAT_OK

**Description**

The function starts a new sawing cycle following a capture event, see chapter 5 „Flying Saw with Hardware Signal“ on page 17.

The function calculates the cutting position by the parameters CapPos and PosOffset. Here, the master position at the capture event is specified by CapPos. PosOffset defines the distance between the capture sensor and the starting position of the sawing axis, see Fig. 23. Please make sure the capture sensor is positioned in a way that it preceeds the sawing axis. The distance is then specified via PosOffset as a negative value.




(1) Starting position of the sawing axis
(2) Addressing position of the sensor
(3) Value range for PosOffset preceeding the starting position: negative
(4) Value range for PosOffset after starting position: positive

**Fig. 23 Distance between the sensor and the starting position**

The distance between the capture sensor and the starting position must be long enough to leave enough time for the sawing axis to synchronize with the cutting position after the function call. This means that the distance between the sensor and the starting position of the sawing axis has to be slightly greater than the acceleration travel of the sawing axis.

**Important** **If too small a distance between capture sensor and starting position of the sawing axis has been chosen, the cut is not carried out at the desired position. If the table function is active, the sawing axis delays with starting the sawing cycle, until the master position has completed a modulo run.**

At each cycle, the material is freshly "referenced" by the capture sensor.

**Important** **If no error has occurred, the function will not be terminated until the sawing axis and the motion master have been synchronized.**

CapPos / PosOffset

The values are related to the position scaling of the motion master.

Processing the Function

The function is called up for each new sawing cycle which specifies the cutting position by a hardware signal.

Handling the Axis Registers

The following axis registers are described for the sawing axis: R101, R402, R410, R443, R446.

**Example**

In the following example, the position of the capture sensor is 400 mm distant from the starting position of the sawing axis in terms of the material running direction.

```
VAR
     ...
     f_CapturePos:                   FLOAT AT %VL 65080;
     ...
VAR_END;

TASK t_Run
     ...
     //--- Carry out the sawing cycles
     WHILE ... DO
         //--- Move the saw to the starting position
             ...
         //--- Wait for the capture event
         //    and store the capture position in f_CapturePos
             f_CapturePos:= ...
             ...
         //--- Start sawing cycle
             MotionFS_StartCapPos(&st_MotionFS_Slave0Data,&st_MotionFS_Slave0TabData0,
                     &f_CapturePos,-400,0,&st_MotionFS_TmpFloatVars0);
         //--- Carry out the cut
             ...
     END_WHILE;
     ...
END_TASK;
```

### 9.8.8 MotionFS_GetStartStatus

**MotionFS_GetStartStatus** (pSlaveData, Option)

**Parameters**

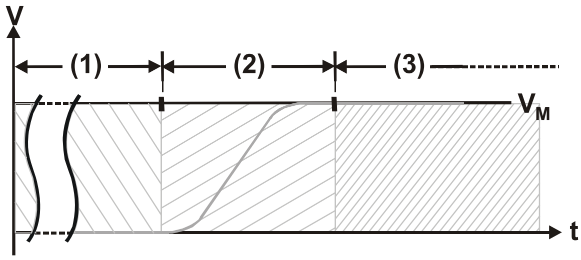
| | |
|---|---|
| pSlaveData: | Start address of the structure, type MOTION_FS_SLAVE_DATA for the sawing axis, for which the starting status is to be determined. |
| Option: | Reserved for expansions: Always = 0 |

**Return Value**

Function Status The possible values are
c_FS_STARTSTATUS_STAT_NO_TABLE_ACTIVE
c_FS_STARTSTATUS_STAT_WAITING_FOR_START
c_FS_STARTSTATUS_STAT_IN_RAMP
c_FS_STARTSTATUS_STAT_SYNCHRON
c_FS_STARTSTATUS_STAT_ERR_INCORRECT_STATUS
Please also refer to chapter 9.6 “Constants” on page 36

**Description**

The function returns the starting status of a sawing cycle, which has already been started by one of the functions MotionFS_Start, MotionFS_StartChangeTab, MotionFS_StartInstant, or MotionFS_StartCapPos.




(1) c_FS_STARTSTATUS_STAT_WAITING_FOR_START = The sawing axis delays, until the master has reached the starting position
(2) c_FS_STARTSTATUS_STAT_IN_RAMP = Sawing axis in ramp
(3) c_FS_STARTSTATUS_STAT_SYNCHRON = The sawing axis is synchronized

**Fig. 24 Status of the Sawing Cycle**

Processing the Function

The function is called up cyclically during the starting phase.

Important **The function only returns a valid status, if it has been called up at the same time as the starting function MotionFS_Start, MotionFS_StartChangeTab, MotionFS_StartInstant oder MotionFS_StartCapPos and during the synchronizing phase until decoupling of the sawing axis.**

Handling the Axis Registers

The function does not describe any axis registers of the sawing axis.

**Example**

In the following example, the starting function is carried out in the task t_Run, while the status function is carried out in the task t_Diag.

```
VAR
     ...
     n_Status:                  INT AT %VL 4;
     ...
VAR_END;

TASK t_Run
     ...
     //--- Carry out the sawing cycles
     WHILE ... DO
         //--- Move the saw to the sawing position
             ...
         //--- Start the sawing cycle
             MotionFS_Start...(...);
         //--- Carry out the cut
             ...
     END_WHILE;
     ...
END_TASK;

TASK t_Diag
     ...
     WHILE TRUE DO   //--- Endless loop
```

```
        //--- Display the status information
        IF ... THEN         //--- Only, if production is in process
            //--- Read the status
                n_Status:= MotionFS_GetStartStatus(&st_MotionFS_Slave0Data, 0);

            //--- Show the status on the display
                CASE n_Status OF
                    c_FS_STARTSTATUS_STAT_NO_TABLE_ACTIVE:
                        //--- Display the status = nought, the display positions are
                            cleared
                            DISPLAY_TEXT (0, 35, '$');
                        BREAK;
                c_FS_STARTSTATUS_STAT_WAITING_FOR_START:
                        //--- Display the status = Wait for the motion master
                            DISPLAY_TEXT (0, 48, 'W');
                        BREAK;
                c_FS_STARTSTATUS_STAT_IN_RAMP:
                        //--- Display the status = in the ramp
                            DISPLAY_TEXT (0, 48, '/');
                        BREAK;
                c_FS_STARTSTATUS_STAT_SYNCHRONOUS:
                        //--- Display the status = synchronous
                            DISPLAY_TEXT (0, 48, '=');
                        BREAK;
                END_CASE;
        END_IF;
    END_WHILE;
    ...
END_TASK;
```

### 9.8.9 MotionFS_GetActCuttingLength

**MotionFS_GetActCuttingLength** (pSlaveData, Direction, Option, pTmp0, pActCuttingLength)

**Parameters**

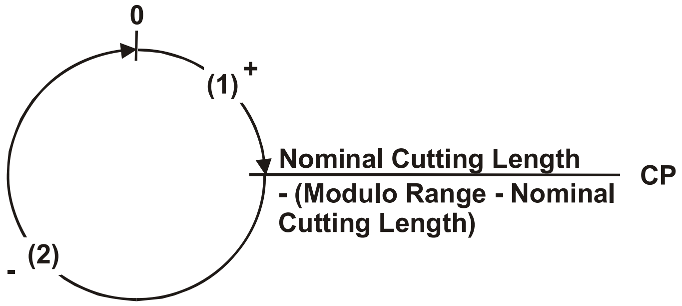
| | |
|---|---|
| pSlaveData: | Start address of the structure, type MOTION_FS_SLAVE_DATA for the sawing axis, for which the length of the material sections is to be determined. |
| Direction: | switch for the direction of material sections counting<br><br>Value range: 0;1<br><br>0 = upcounting<br>1 = downcounting |
| Option: | Reserved for expansions: Always = 0 |
| pTmp0: | Start address of the structure of the temporary FLOAT-variable, type MOTION_FS_TMP_FLOATVARS. |
| pActCuttingLength: | FLOAT return value: Address of the FLOAT variable containing the actual length of the material sections in mm respectively °; it is calculated by the function.<br><br>Value range of the length of material sections: see below |

**Return Value** No value returned by the function itself, yet see parameter pActCuttingLength

**Description**

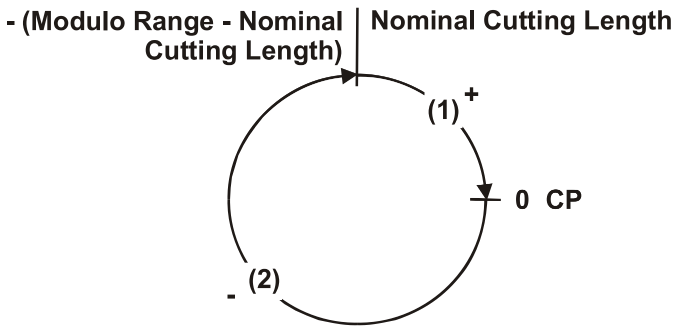
During the starting phase of the sawing cycle of a classical Flying Saw, this function feeds back the already accumulated length of material. The accumulating phase is the time when the sawing axis is waiting for the starting position of the motion master and when it is travelling the synchronising ramp, until it has been synchronized with the motion master. In this process, this function is executed in a separate task parallel to the starting function MotionFS_Start or MotionFS_StartChangeTab.

The value range of the feedback parameter ActCuttingLength can be taken from the following illustrations.




(CP) Cutting position
Nominal cutting length
Modulo position range
(1) Accumulating phase: increasing from 0 to nominal cutting length
(2) Cutting phase: decreasing from - (modulo position range – nominal cutting length) to 0

**Fig. 25 Present material length – counting upward**




(CP) Cutting position
Nominal cutting length
Modulo position range
(1) Accumulating phase: decreasing from the nominal cutting length - 0
(2) Cutting phase: decreasing from 0 to - (modulo position range – nominal cutting length)

**Fig. 26 Present material length – counting downward**

In order to stop the value from increasing as soon as the nominal cutting length has been reached, in order to output only the nominal cutting length, for example, the feedback parameter of the function can be queried for < 0. If this is the case, the sawing axis is in the cutting phase; thus, the nominal cutting length has been reached. For this, please refer to the example regarding the function.

pActCuttingLength

The material length is related to the position scaling of the motion master.

Processing the Function

The function is called up cyclically during the starting phase.

**Important**

**The function only feeds back a valid value, if it is called up parallel to the already running starting function MotionFS_Start or MotionFS_StartChangeTab and during the synchronous phase.**

**The precise time interval for function execution is defined as follows: From issuing command 46 via start function (return value of the function MotionFS_GetStartStatus > c_FS_STARTSTATUS_STAT_NO _TABLE_ACTIVE) up to disengaging.**

Handling the Axis Registers

The function does not describe any axis registers of the sawing axis.

Troubleshooting

If the fed back material length value does not make sense, the reason may be that the set cutting length can neither be achieved at the present speed of the motion master nor by accelerating the sawing axis.

**Example**

In the following example, the starting function is carried out in the task t_Run, while the function MotionFS_GetActCuttingLength is carried out in the task t_Diag. The material length is to be counted up.

```
VAR
      ...
      f_ActCuttingLength:     FLOAT AT %VL 65071;
      ...
VAR_END;


TASK t_Run
      ...
      //--- Carry out the sawing cycles
      WHILE ... DO
          //--- Move the saw to the starting position
              ...
          //--- Start the sawing cycle
              MotionFS_Start...(...,f_CuttingLength,...);
          //--- Carry out the cut
              ...
      END_WHILE;
      ...
END_TASK;
 TASK t_Diag
      ...
      WHILE TRUE DO   //--- Endless loop
          //--- Display the present material length,
          //    Yet only, if table function is active
          IF MotionFS_GetStartStatus(&st_MotionFS_Slave0Data,0) >
                                      c_FS_STARTSTATUS_STAT_NO_TABLE_ACTIVE THEN
              MotionFS_GetActCuttingLength(&st_MotionFS_Slave0Data,0,0,
                                      &st_MotionFS_TmpFloatVars1,&f_ActCuttingLength);
              //--- If in the cutting phase, f_ActCuttingLength becomes negative.
              //    In this case, the value shown on the display is to present the
              //    set cutting length. If the present material length >
              //    is the set cutting length, negative values up to -500 max.
              //    are to be permitted.
                  IF f_ActCuttingLength < -500 THEN
                      f_ActCuttingLength:= f_CuttingLength;
                  END_IF;
              //--- Show on the display
                  DISPLAY_VALUE (0, 35, f_ActCuttingLength);
              ...
          END_IF;
      END_WHILE;
      ...
 END_TASK;
```

### 9.8.10 MotionFS_GetDistToCutPoint

**MotionFS_GetDistToCutPoint** (pSlaveData, Option, pTmp0, pDistCutPoint)

**Parameters**

| | |
|---|---|
| pSlaveData: | Start address of the structure, type MOTION_FS_SLAVE_DATA for the sawing axis, for which the travel distance is to be determined. |
| Option: | Reserved for expansions: Always = 0 |

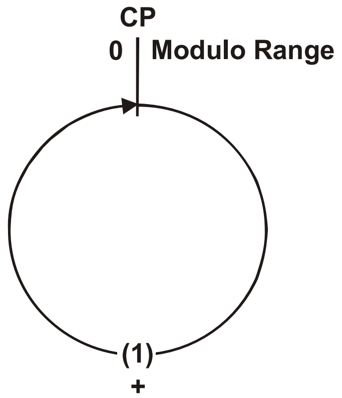
| | |
|---|---|
| pTmp0: | Start address of the structure of the temporary FLOAT-variable, type MOTION_FS_TMP_FLOATVARS. |
| pDistCutPoint: | FLOAT return value: Address of the FLOAT variable containing the actual travel distance of the material sections in mm respectively °; it is calculated by the function, see function description.<br><br>Value range of the travel distance: see below |

**Return Value** No value returned by the function itself, yet see parameter pDistCutPoint

**Description**

During the starting phase of the sawing cycle, the present distance from the cutting position is fed back by this function. Simultaneously to the presently applied starting function, this function is carried out in a separate task. This function can be used for all starting functions (MotionFS_Start, MotionFS_StartChangeTab, MotionFS_StartInstant oder MotionFS_StartCapPos).

The value range of the return value parameter DistCutPoint can be taken from the following illustrations. The value range is always within the positive range of numbers.




(CP) Cutting position
Modulo position range
(1) descending: from the modulo position range to 0

**Fig. 27 Distance to the cutting position**

pDistCutPoint

The distance is related to the position scaling of the motion master.

Processing the Function

The function is called up cyclically during the starting phase.

**The function only feeds back a valid value, if it is called up parallel to the already running starting function and during the synchronous phase.**

**Important**

**The precise time interval for function execution is defined as follows: From issuing command 46 via start function (return value of the function MotionFS_GetStartStatus > c_FS_STARTSTATUS_STAT_NO
_TABLE_ACTIVE) up to disengaging.**

Handling the Axis Registers

The function does not describe any axis registers of the sawing axis.

Troubleshooting

If the returned distance value does not make sense, the reason may be that the set cutting length can neither be achieved at the present speed of the motion master nor by accelerating the sawing axis.

**Example**

In the following example, the starting function is carried out in the task t_Run, while the function MotionFS_GetDistToCutPoint is carried out in the task t_Diag.

```
VAR
      ...
      f_ActCuttingLength:     FLOAT AT %VL 65071;
      ...
VAR_END;


TASK t_Run
      ...
      //--- Carry out the sawing cycles
      WHILE ... DO
          //--- Move the saw to the starting position
              ...
          //--- Start the sawing cycle
              MotionFS_Start...(...,f_CuttingLength,...);
          //--- Carry out the cut
              ...
      END_WHILE;
      ...
END_TASK;

 TASK t_Diag
      ...
      WHILE TRUE DO   //--- Endless loop
          //--- Display the present distance to the cutting position
          //    Yet only, if table function is active
          IF MotionFS_GetStartStatus(&st_MotionFS_Slave0Data, 0) >
                                 c_FS_STARTSTATUS_STAT_NO_TABLE_ACTIVE THEN
              MotionFS_GetDistToCutPoint(&st_MotionFS_Slave0Data,0,
                                 &st_MotionFS_TmpFloatVars1,&f_ActCuttingLength);
              //--- Show the distance to the display
                  DISPLAY_VALUE (0, 35, f_ActCuttingLength);
              ...
          END_IF;
      END_WHILE;
      ...
 END_TASK;
```

### 9.8.11 MotionFS_GetMasterAlignPos

**MotionFS_GetMasterAlignPos** (pSlaveData, pTabData, Option, pTmp0, pMasterAlignPos)

**Parameters**

| | |
|---|---|
| pSlaveData: | Start address of the structure, type MOTION_FS_SLAVE_DATA for the sawing axis, which is to be aligned manually. |
| pTabData: | Start address of the structure of the type MOTION_FS_TAB_DATA containing the data of the table by which the new sawing cycle is to be started. |
| Option: | Reserved for expansions: Always = 0 |
| pTmp0: | Start address of the structure of the temporary FLOAT-variable, type MOTION_FS_TMP_FLOATVARS. |
| pMasterAlignPos: | FLOAT return value: Address of the FLOAT variable containing the actual alignment position of the material sections in mm respectively °; it is calculated by the function, see function description.<br><br>Value range of the alignment position: Master position range |

**Return Value** No value returned by the function itself, yet see parameter pMasterAlignPos

**Description**

The function is used for manually "referencing" the material by means of the tools unit. It reports back the position the material (motion master) must have in order to place the first cut correctly. For this, the cutting position on the material and the starting position of the sawing axis must be on the same level. Please refer to Fig. 11 in chapter 4.2“ Variant 2: Starting by Adjusting Material and Tools Unit“ on page 14.

**The return value is only valid, if the production period is also started by means of the table that has been defined for carrying out the function.**

**Important**

For alignment, please proceed as follows:

- Cutting position on the material is aligned to the height of the starting position, e.g. by manually operating various keys. The step accuracy that is to follow depends on this alignment among other parameters.
- The position of the motion master is set to the return value of the function.

<u>pMasterAlignPos</u>

The alignment position is related to the position scaling of the motion master.

<u>Processing the Function</u>

For a classical Flying Saw, this function is to precede a production period, if there is no head cut to be carried out.

<u>Handling the Axis Registers</u>

The function does not describe any axis registers of the sawing axis.

**Example**

In the following example, the production period is started by means of table 0.

```
VAR
     ...
     f_AlignPos:             FLOAT   AT %VL 65078;
     ...
VAR_END;


TASK t_Run
     ...
     //--- Reference (set) the motion master to the reference position.
     //    For this, both the material and the tools unit have been adjusted manually
         //    at the starting position of the sawing axis
         MotionFS_GetMasterAlignPos(&st_MotionFS_Slave0Data,&st_MotionFS_Slave0TabData0,
                             0,&f_AlignPos,&st_MotionFS_TmpFloatVars0);
         MotionHome ( Ax, Grundstellung setzen, f_AlignPos);

         //--- Carry out the sawing cycles
         WHILE ... DO
         //--- Move the saw to the starting position
             ...
         //--- Start the sawing cycle
             MotionFS_Start(...,f_CuttingLength,...);
         //--- Carry out the cut
             ...
     END_WHILE;
     ...
END_TASK;
```

### 9.8.12 MotionFS_SetSyncCorrection

**MotionFS_SetSyncCorrection** (pSlaveData, pCorrection, Option)

**Parameters**

| | |
|---|---|
| pSlaveData: | Start address of the structure, type MOTION_FS_SLAVE_DATA for the sawing axis, which is to be aligned. |
| pCorrection: | Address of the FLOAT variables containing the change value for the adjustment of the synchronization.<br><br>Value range of the correction: FLOAT range |
| Option: | Reserved for expansions: Always = 0 |

**Return Value** No return value

**Description**

The function adjusts the synchronization of the sawing axis with the motion master by the value being transmitted by the "Correction" parameter. For this, please refer to 7.3 „Adjustment of Synchronous Operation“ on page 26.

The function adds the value of the parameter "Correction" to the parameter SyncFactor of the structure MOTION_FS_SLAVE_DATA, see chapter 9.5.1 „MOTION_FS_SLAVE_DATA“ on page 33. At calling up a starting function, the value of the parameter "SyncFactor" is directly written into axis register R446 "Table – Slave Scaling".

**Important** **The alignment does not become effective before the next sawing cycle. Further, the cutting position is shifted as from the moment of adjustment.**

**Important** **Synchronization adjustment by means of this function is only applied during commissioning, in order to determine the correction factor. During production, synchronization is adjusted in the motion master and the sawing axis by means of the gearbox factors.**

Processing the Function

The function is processed at commissioning the synchronizing adjustment before calling up a start function.

Handling the Axis Registers

The function does not describe any axis registers of the sawing axis.

**Example**

In the following example, synchronizing is adjusted to a factor of –0.2; this means that after the adjustment, the sawing axis is 0.2 times slower than before.

```
VAR
     ...
     f_SyncFactorCorr:   FLOAT AT %VL 65081;
     ...
VAR_END;


TASK t_Run
     ...
     //--- Synchronisation adjustment by factor 0.2
         IF ...Then //--- only, if adjustment is required
             f_SyncFactorCorr:= -0.2;
             MotionFS_SetSyncCorrection(&st_MotionFS_Slave0Data,&f_SyncFactorCorr,0);
         END_IF;

     //--- Carry out the sawing cycles
     WHILE ... DO
         //--- Move the saw to the starting position
             ...
         //--- Start the sawing cycle
             MotionFS_Start(...,f_CuttingLength,...);
         //--- Carry out the cut
             ...
     END_WHILE;
     ...
END_TASK;
```