MX Driver Interfaces

```
typedef struct
        mx_status_type ( *motor_is_busy )( MX_MOTOR *motor );
        mx_status_type ( *move_absolute )( MX_MOTOR *motor );
        mx_status_type ( *get_position )( MX_MOTOR *motor );
        mx_status_type ( *set_position )( MX_MOTOR *motor );
        mx_status_type ( *soft_abort )( MX_MOTOR *motor );
        mx_status_type ( *immediate_abort )( MX_MOTOR *motor );
        mx_status_type ( *positive_limit_hit )( MX_MOTOR *motor );
        mx_status_type ( *negative_limit_hit )( MX_MOTOR *motor );
        mx_status_type ( *find_home_position )( MX_MOTOR *motor );
 MX_MOTOR_FUNCTION_LIST;
```

```
typedef struct
        mx_status_type ( *clear ) ( MX_SCALER *scaler );
        mx_status_type ( *overflow_set ) ( MX_SCALER *scaler );
        mx_status_type ( *read ) ( MX_SCALER *scaler );
        mx_status_type ( *read_and_clear ) ( MX_SCALER *scaler );
        mx_status_type ( *write ) ( MX_SCALER *scaler );
 MX_SCALER_FUNCTION_LIST;
```

Example Program

```
  ... some initialization code ...
energy_motor = mx_get_record( record_list, "energy" );
i_zero       = mx_get_record( record_list, "I0" );
i_trans      = mx_get_record( record_list, "It" );
timer1       = mx_get_record( record_list, "timer1" );
start = 8950; step_size = 1.0; num_steps = 101;
integration_time = 1.0;
for ( i = 0; i < num_steps ; i++ ) {
        mx_scaler_clear( i_zero );
        mx_scaler_clear( i_trans );
        energy = start + step_size * (double) i;
        mx_motor_move_absolute( energy_motor,
                                    energy, NULL, 0 );
        mx_timer_start( timer1, integration_time );
        busy = TRUE;
        while ( busy ) {
                mx_timer_is_busy( timer1, &busy );
                mx_msleep(1);
        }
        mx_scaler_read( i_zero, &i_zero_value );
        mx_scaler_read( i_trans, &i_trans_value );
        printf( "%10.3f  %10lu  %10lu\n",
                energy, i_zero_value, i_trans_value );
}
```